IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 3, 2018, accepted August 15, 2018, date of publication August 20, 2018, date of current version September 21, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2866082

The Empirical Study of Semi-Supervised Deep Fuzzy C-Mean Clustering for Software Fault Prediction

ALI ARSHAD[1,2], SAMAN RIAZ[1,2], LICHENG JIAO[3], (Fellow, IEEE), AND APARNA MURTHY[4]

[1]School of Computer Science and Technology, Xidian University, Xi'an 710071, China
[2]School of International Education, Xidian University, Xi'an 710071, China
[3]Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China, International Joint Collaboration Laboratory of Intelligent Perception and Computation, International Research Center of Intelligent Perception and Computation, School of Artificial Intelligence, Xidian University, Xi'an 710071, China
[4]Professional Engineers in Ontario, North York, ON M2N 6K9, Canada

Corresponding author: Ali Arshad (alli.arshad@gmail.com)

This work was supported in part by the National Basic Research Program (973 Program) of China under Grant 2013CB329402, in part by the National Natural Science Foundation of China under Grant 61573267, Grant 61473215, Grant 61571342, Grant 61572383, Grant 61501353, Grant 61502369, Grant 61271302, Grant 61272282, and Grant 61202176, in part by the Fund for Foreign Scholars in University Research and Teaching Programs (the 111 Project) under Grant B07048, and in part by the Major Research Plan of the National Natural Science Foundation of China under Grant 91438201 and Grant 91438103.

ABSTRACT Software fault prediction is a very consequent research topic for software quality assurance. The performance of fault prediction model depends on the features that are used to train it. Redundant and irrelevant features can hinder the performance of a classification model. In this paper, we propose an empirical study of two-stage data pre-processing technique on software fault prediction models. In the first stage, a novel semi-supervised deep Fuzzy C-Mean (DFCM) clustering-based feature extraction technique is proposed to create new features by utilizing deep multi-clusters of unlabeled and labeled data sets that tends to maximize intra-cluster class and intra-cluster feature by using FCM clustering. The FCM also utilizes to handle the class imbalance problem. In the second stage, we further ameliorate the prediction performance with coalescence of feature selection (using random-under sampling) to reduce the noisy data for classification. However, by the performance of the model results in the amalgamation of novel DFCM data pre-processing approach work better due to their ability to identify and amalgamation essential information in data features. An empirical study is designed on real-world software project (NASA & Eclipse) data set to evaluate the performance of DFCM by implemented different data pre-processing schemes on prediction models (C4.5, naive bayes, and 1-near neighbor (1-NN)), which are widely used in software fault prediction and further investigated the influencing factors in our approach. The result shows that the performance of the proposed DFCM feature extraction technique for data pre-processing is stable and effectiveness on all prediction models.

INDEX TERMS Semi-supervised learning, Fuzzy C-Mean clustering, feature learning, software fault prediction.

I. INTRODUCTION

Software fault prediction (SFP) is one of the hottest research topics in experimental software engineering to test reliability and quality of software entities. The complexity and size of software are rapidly increasing day by day for sundry reasons incrementing authoritative ordinance of infusion of incipient technologies, reliability, and security by the users. One possible way to handle this issue is to focus on determination of redundant preprocessing data in early phases of software fault prediction. Software defect prediction can be regarded as the binary classification problem, which aims to divide software modules into fault modules or non-fault modules. To improve software quality assurance, many researchers paid attention to the process of testing and withal on preprocessing training data [1]–[11], [41], [42].

Supervised learning models are one of the best choices for software fault prediction if labeled data are provided for training model [12], [13]. Consequently, for the better

accuracy one drawback of supervised learning is that the size of training dataset should be as large as possible, which is expensive and time-consuming. Supervised learning models use class labeled data represented as known fault data during the training phase. However, there are cases when previous known faults data are not inordinate available for training model, then to handle this challenging problem, the semi-supervised approach can be applied in these case. Fig. 1 is a semi-supervised learning approach because in the training phase uses labeled data represented as known faults data and unlabeled data represented as unknown faults data.

Many researchers have proposed the semi-supervised approach for classification [14], [15], [16]. However, few researchers have been used simultaneously to exploit the obnubilated information from unlabeled data to labeled data [17]–[20], [42]. Many semi-supervised approaches are used for fault prediction. However, most of them have been dealing with balanced classes [13].

Real-world datasets are usually imbalanced, causes a difficulty for most traditional classification algorithms, example decision tree, Naive Bayes, support vector machine, random forest and evolutionary algorithms [6], [21]–[25], [27], [28]. The cause of imbalanced problem occurs where some classes are highly underrepresented compared to other classes. Particularly, for a binary class application, level of imbalance is defined as the ratio of the number of majority examples to that of the minority ones, and the minority class is always more interest [29], [30]. This problem affects the performance of the classification models. This problem has gained more attention from researchers, lately. The K-nearest neighbor classifier (KNN) [31], [32] is one of the most popular learning algorithms for imbalance classes. An object is assigned to the class which is most frequent among the K-nearest neighbor. At the time, numerous changes in KNN have been proposed for improvement [33], [34]. There are many Fuzzy set theory based algorithms [35]–[38] are proposed for imbalanced classes.

Nevertheless, the imbalanced dataset is not the only factor that harms the performance of classifiers. High dimensionality datasets in training set may cause lead to a high computational (training) cost and performance degradation of certain classification models [39], [40] by eliminating the feature which is mostly irrelevant to the class.

In this paper, we present an empirical study to evaluate the performance stability of our previous work preprocessing raining data technique [42] on traditional classification models with class imbalance. In our empirical study, the aim to check the effectiveness of the new semi-supervised approach in which both supervised and unsupervised data are utilized simultaneously during the multi-clustering process in which the obnubilated information is exploited from unlabeled data to support the construction of better classifier. The coalesce analysis of labeled data and unlabeled data is very useful in the field of pattern recognition.

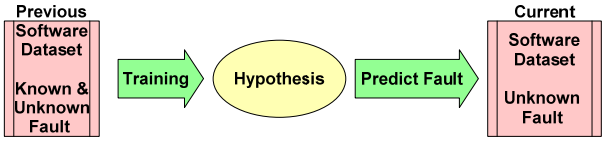

FIGURE 1. The process of software fault prediction.

However, to the best of our knowledge very few researchers have utilized multi-clustering for feature extraction and also handle the class imbalance problem [43], in which Germain Forestier, proposed a semi-supervised learning method to produce new features derived from the first step of data clustering by utilizing supervised and unsupervised data. They used unsupervised classification to create new features to describe the labeled samples by creating clusters that tend to maximize intra-cluster similarity and intra-cluster dissimilarity.

However, the performance of accuracy effect with class imbalance problem. The prosperity of prediction model depends strongly on the features and it is commonly believed, more features do not indispensably avail identification of systems based on input-output data. Hence, there are two common ways for dimensionality reduction of features, one is by feature extraction [44], [45] and second by feature selection [46], [47]. However, few researchers have cumulated these two ways of dimension reduction of features to ameliorate data quality in software fault prediction.

In this paper, we used ''Deep'' word in our paper title, because to create new features by deep correlation between supervised and unsupervised data during the multi-clusters for feature extraction.

The contribution of the proposed method can be concluded as follow.

1. We proposed semi-supervised deep FCM clustering data pre-processing approach to create new features, which simultaneously deals with the labeled and unlabeled data during clustering to exploit the hidden information from unlabeled data.
2. We introduced semi-supervised FCM multi-clustering that tend to maximize intra-cluster class and intra-cluster features to handle the class imbalance problem.
3. An Empirical study is designed to evaluate the performance stability of proposed approach with different schemes on three prediction models. The result shows that the performance of Navie Bayes is more stable than other prediction models on our proposed approach.

In this study, we design experiments to analyze the impact of features extraction by DFCM on the performance of commonly used classification techniques (Naive Bayes (NB), C4.5, 1-NN). We train our classification models using dataset based on real-world software projects to demonstrate the effectiveness of our approach.

We conduct an empirical study to evaluate the performance stability of our approach based on the address the following research questions.

RQ1. In our feature extraction stage, whether FCM based semi-supervised multi-clustering can improve the performance of the classification models with imbalanced classes?

RQ2. Does combining the feature selection technique with our feature extraction technique together have more significantly improved than by using any of the individual prediction performance?

RQ3. Does combining RUS approach with DFCM together have a more significant improvement in prediction performance?

This paper is organized as follows. In section II, we will provide a review of related work, Section III, deals with implementation strategy of the algorithm, Section IV, describes experiment and results, section V, provides with the threats to validity and section VI, provides with the conclusion.

II. RELATED WORK

Software fault prediction is one of the most consequential tasks to predict the software modules. Many researchers have used a variety of machine learning technique (such as k nearest neighbor rule decision, clustering, and SVM [5], [7], [8], [48] for classification model [13], [49]–[53] to categorize software into fault and non-fault.

Semi-Supervised learning plays an important role to improve the performance of the model in the machine learning techniques. Semi-supervised approach for software fault prediction is studied by yarowsky, the algorithm utilized in speech processing computational linguistics [54]. Here, a set of untagged data is used and steps of collection labeling using this labeled data are trained for partitioning, iteratively on the probability of co-occurrence till the data grows and reduces the untagged set. Once the grouping is complete, the classifier is used. The algorithm is dependent on the collocational list of entries [54].

Lu *et al.* [55] invested the performance of an iterative semi-supervised software defect prediction approach on different size of labeled rate, they approved if the rate of labeled data is greater than 5% then the proposed approach performs better than supervised learning approach.

Data pre-processing is valuable to amend the software data quality [51], [56], [57] which includes feature selection and re-sampling. Feature selection is used to remove the irrelevant feature from the dataset, which will hurt the generalization performance of classification [58]. Feature rank algorithm [3], [56], [59] are used for selection on the bases of importance weights in differentiating modules at different classes [3], [60]–[62].

Liu *et al.* proposed a two-stage data preprocessing approach for software fault prediction [11]. It is a two-stage data preprocessing approach, which integrates both feature selection and instance reduction, to improve the quality of software fault prediction. He proposed NTC (NB) (Novel threshold-based clustering algorithm using Naïve Bayes classification model), which involves both reliance analysis and redundancy control. He also applies the random under-sampling technique to keep the balance between the faulty and non-faulty classes.

In feature learning Coates [63], apply several off-the-shelf feature learning algorithms, by the analysis of this results the clustering algorithms is extremely fast and easy to implement with achievable high accuracy.

It is known that good modeling tool is the cognition between a learning method and feature representation learning. Dimensionally reduction in feature learning is usually done in two broad ways by feature extraction (generation of incipient features from subsisting ones) [44], [45] and features selection [46], [47]. The data preprocessing plays an important role to improve the quality of software datasets [51], [56], [61], which include feature selection and reduction (or sampling). Khoshgoftaar *et al.* [64] combined filter based feature ranking methods and random under-sampling for improved the data preprocessing.

Many unnecessary features are one of the causes of high dimensionality problem, and the other is class imbalance problem. In recent years, the class imbalance is one of the most important tasks for software fault prediction. In order to explore the impact of class imbalance on SFP, many researchers have carried out a large number of empirical studies [65]–[68]. Wang *et al.* [65] provided an empirical study on bases of stability of feature selection techniques. By their experimental results, many factors could affect the stability of feature selection, such as class balance, feature subset size, and perturbation level. Grbac *et al.* [67] also indicated in their investigation that feature selection was unstable with the higher level of data imbalance.

Yu *et al.* [69] investigate the performance stability of fault prediction models with class imbalance on six prediction models (C4.5 [21], Naive Bayes [22], KNN [70], Logistic Regression [71], Multi-layer Perceptron (MLP) [72], and Random Forest [23]. The experiment results showed that the performance of C4.5 is unstable on the imbalanced dataset, and the performance of Naive Bayes and Random Forest are more suitable than other models.

According to our knowledge, there have few researchers used fuzzy set theory to software fault prediction been a few attempts to use fuzzy set theory to predict software faults. Pandey and Goyal [73] first constructed a decision tree using ID3 and then from decision tree they generate ''if-then rules, which are used as fuzzy rules''. Chatterjee and Mayi [74] also use fuzzy if the fault in software requirement analysis phase.

Li [75] proposed Constraint FCM method novel semi-supervised fuzzy c-means algorithm. It uses data that contain labeled tag and finds cluster center and optimize the objective function of fuzzy c-mean of the labeled data using EM algorithm.

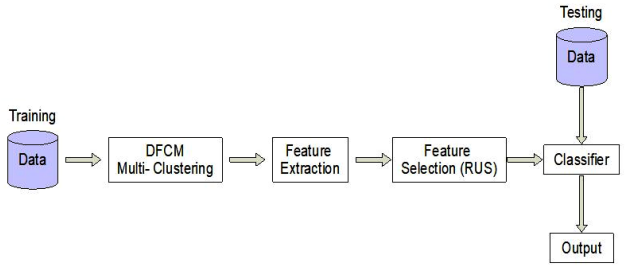

FIGURE 2. The framework of proposed approach.

However, to the best of our knowledge, very few researchers have used the semi-supervised approach in multi-clusters for feature selection to handle the class imbalance problem. Forestier and Wemmert [43] proposed a semi-supervised learning method to produce new feature by multi-clusters.

Gabrys and Petrakieva [76] or Bouchacha [77], they proposed the method to ameliorate the classification accuracy with very few labeled and abundant unlabeled samples are available, they used the semi-supervised approach in which during the clustering process they deal with labeled and unlabeled data simultaneously.

Cai [19] proposed "A simultaneous learning framework for clustering and classification" to fuse the advantages of classification learning and clustering learning into the single framework with inhibited labeled and abundant unlabeled data by optimizing the clustering centers in the objective function, both the classification learning and clustering learning can be realized simultaneously. In his work, they used an evolutionary technique called modified particle swarm optimizer (PSOm) to find optimal clustering centers.

III. EMPIRICAL STUDY OF SEMI-SUPERVISED DEEP FUZZY-C MEANS CLUSTERING

In this section, we present the empirical study to evaluate the performance of our previous work "Semi-Supervised Deep Fuzzy C-Mean Clustering for Software Fault Prediction" [42]. In semi-supervised deep fuzzy C-Mean clustering, we present a human-interpretable learning-based semi-supervised feature extraction technique for software fault prediction model, which cumulates the DFCM multi-clustering based feature extraction with feature selection (random-under sampling (RUS) [23]) technique for fault prediction models. We count the widely used prediction models in software fault prediction, C4.5 [21] is a decision tree algorithm, 1-NN [78] is an instance based algorithm and NB [22] is a probabilistic classifier based on Bayes theorem, and it supposes that all features are independent.

A. THE FRAMEWORK OF OUR APPROACH

Fig. 2 gives the framework of our approach for fault prediction. In this framework, we cumulate the multi-clustering based feature extraction with random-under sampling [23] for fault prediction model. Fig. 3 is the flowchart of DFCM clustering based feature extraction to remove the irrelevant feature with class imbalance. In this flowchart, after normalization of semi-supervised data using minimum-maximum approach, data converted into binary clusters of labeled and unlabeled datasets, which lead to intra-clusters of classes and feature. We have total k (2+1) clusters in the last stage of clustering, where k is the number of features. Calculate the DFCM membership and DFCM centroid by algorithm 1.

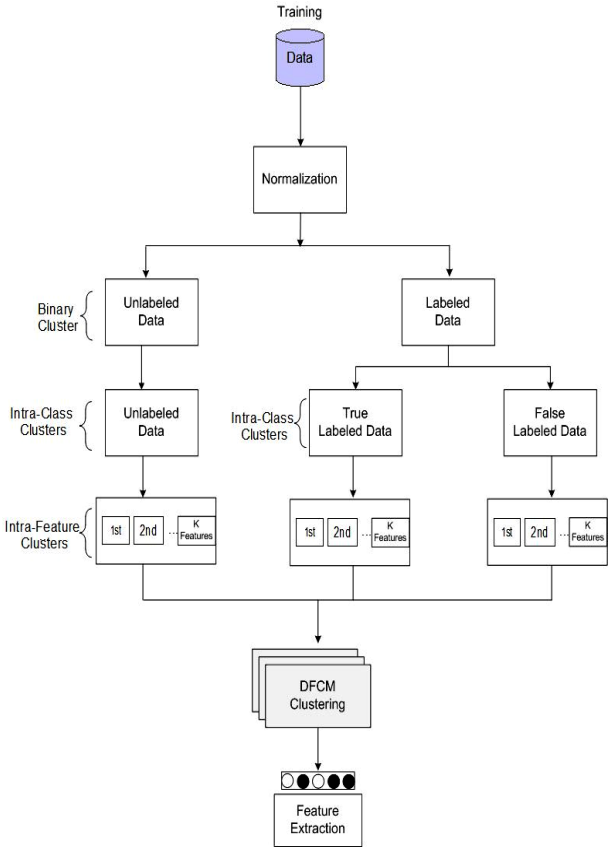

FIGURE 3. Flowchart of DFCM clustering based feature extraction.

B. FEATURE EXTRACTION

Use of many features customarily increases the data acquisition cost and time. Therefore, it is always desirable for classification that the number of features reduces the accumulated the design for decision-making system. There are two main broad ways to reduce the feature space i.e. feature selection [38], [47] and feature extraction [44], [45].

But in our proposed method, we used both methods to design good prediction system by generating good features and removing irrelevant and redundant features to reduce the noisy data for training classification model.

In the next stage for feature extraction on the bases of DFCM k (2+1) clusters, we apply we apply DFCM clustering to learn k centroids from the labeled and unlabeled datasets. Given the learned centroids $V^{(k)}$, we choose non-linear mapping for feature mapping.

$$f_k(x) = \max(0, \mu(z) - z_k) \quad (1)$$

Algorithm 1 Membership and Centroid of DFCM

Input:

The data set $X = \{x_1, x_2 \ldots .., x_n, l\}$,
$X = X_{TL} \cup X_{FL} \cup X_{UN}$, where
$X_{TL} = \{x_1, x_2 \ldots .., x_T\} \in$ *True* class,
$X_{FL} = \{x_{T+1}, x_{T+2} \ldots .., x_l\} \in$ *False* class,
$l \in$ *True*, *False*,
$X_{UN} = \{x_{F+1}, x_{F+2} \ldots .., x_n\} \in$ *Unlabeled* class,
Where l is the labeled classes and k is the feature clusters, fuzziness m=2, with ε is objective threshold and t is number of iterations.

Output:

$U_{TL}, U_{FL} \& U_{UN}$ Membership matrices
$V_{TL}^{(k)}, V_{FL}^{(k)} \& V_{UN}^{(k)}$ Set of k centroid.

1. Initialize the sets of cluster center $V_{TL}^{(o)}, V_{FL}^{(o)} \& V_{UN}^{(o)}$ by randomly select "k" cluster centers from the features of $X_{TL}, X_{FL} \& X_{UN}$ datasets respectively.
2. Construct membership matrices $U_{TL}, U_{FL} \& U_{UN}$ with random decimal fraction.
3. Compute cluster center $V_{TL}^{(k)}, V_{FL}^{(k)} \& V_{UN}^{(k)}$ using formula of cluster center of FCM [32].
4. Update $U_{TL}, U_{FL} \& U_{UN}$ using formula of membership of FCM [37].
5. Repeat step 2 & 3 until $\left\| \mathrm{J}^{(t)} - \mathrm{J}^{(t-1)} \right\| < \varepsilon$ for all labeled and unlabeled subsets separately.

Where $z_k = \left\| x - V^{(k)} \right\|_2$ and $\mu(z)$ is the mean of the elements of z. If the output 0 of any feature f_k, where the distance to the centriod V^k is "above average". In practice, this means that roughly half of the feature will be set 0.

After the feature extraction by algorithm 2, to balance the number of features between all subsets of labeled classes and unlabeled dataset, select "s" features from each subset by feature selection RUS (Random under-sampling) suggested by Khoshgoftaar *et al.* [64], where "s" is the number of minimum features in any subset shown in Table 3.

IV. EXPERIMENT

In this section, we design experiments to conduct an empirical study to evaluate the performance of our approach Deep Fuzzy C-Mean (DFCM) clustering based feature extraction with Random-under Sampling (RUS) on fault prediction models. First, we design the research questions for empirical study, second, we describe the details of the dataset used in our experiments, third, we introduced the performance measuring for evaluation and in the last, and we design experiments on the bases of research questions.

A. RESEARCH QUESTIONS

We train our fault prediction models using a combination of DFCM clustering based feature extraction and feature selection techniques, in order to address the following research questions.

RQ1. In our feature extraction stage, whether FCM based semi-supervised multi-clustering can improve the performance of the classification models with imbalanced classes?

RQ2. Does combining the feature selection technique with our feature extraction technique together have more significantly improved than by using any of the individual prediction performance?

RQ3. Does combining RUS approach with DFCM together have a more significant improvement in prediction performance?

Algorithm 2 Feature Extraction of DFCM

Input:

The data set $X = \{x_1, x_2 \ldots .., x_n, l\}$,
$X = X_{TL} \cup X_{FL} \cup X_{UN}$,
Set of centroid $V_{TL}^{(k)}, V_{FL}^{(k)} \& V_{UN}^{(k)}$

Output:

$f_{TLk}(x_{TL}), f_{FLk}(x_{FL}) \& f_{UNk}$, sets of features of True class, False class and unlabeled dataset.

1. Calculate $Z_{TLk}, Z_{FLk} \& Z_{unk}$, using formula
   Where,
   $$Z_{TLk} = \left\| x_{TL} - V_{TL}^{(k)} \right\|,$$
   $$Z_{FLk} = \left\| x_{FL} - V_{FL}^{(k)} \right\| \&$$
   $$Z_{UNk} = \left\| x_{UN} - V_{UN}^{(k)} \right\|.$$
2. Calculate $\mu_T(Z_{TL}), \mu_F(Z_{FL}) \& \mu_T(Z_{UN})$ are the means of the elements $Z_{TL}, Z_{FL} \& Z_{UN}$.

1) $f_k(x) = \max(0, \mu(z) - z_k) \forall TL, FL \& UN$ *features*
2) Update all the features of all data sets $f_{TLk}, f_{FLk} \& f_{UNk}$.

B. DATA PREPARATION

In this paper, the MATLAB 2017a [79] platform is utilized to test the results. In order to evaluate the performance of our model to test the experiment on thirteen datasets, ten datasets belong to NASA (cm1, jm1, kc1, kc3, mc2, mw1, pc1, pc3, pc4, and pc5) [80], [81], and three datasets belong to Eclipse (Eclipse 2.0, Eclipse 2.0 and Eclipse 3.0) [82], [83] with 10%, 20%, &30% rate of labeled data. All datasets have binary classes with imbalance ratio. To stop the iteration for updating new cluster center for all datasets, we set objective threshold 0.1, and degree of fuzziness m=2.

Table 1, shows the benchmark NASA and Eclipse datasets that illustrates brief properties of thirteen datasets, that will include the number of samples, number of features, number of faulty modules, number of non-faulty modules, and number of classes.

C. PERFORMANCE MEASURE

There are many ways how to measure well statistical model predicts a binary outcome with imbalanced classes. We used three most common measures to estimate the performance of our approach by Sensitivity, Specificity, and Area-under-the-curve (AUC) by using ROC-curve under the information of

TABLE 1. NASA and eclipse dataset.

Dataset	No. Of Samples	No. Of Features	No. Of Faulty Modules	No. of Non-Faulty Modules	Imbalance Rates
Eclipse 2.0	6729	155	975	5714	5.86
Eclipse 2.1	7888	155	854	7034	8.24
Eclipse 3.0	10593	155	1568	9025	5.76
cm1	327	37	42	285	6.79
jm1	7782	21	1672	6110	3.65
kc1	2109	20	326	1783	5.50
kc3	194	39	36	158	4.39
mc2	125	39	44	81	1.84
mw1	253	37	27	226	8.37
pc1	705	37	61	644	10.56
pc3	1077	37	134	943	7.04
pc4	1458	37	178	1280	7.19
pc5	17186	38	516	16670	32.31

TABLE 2. Fault prediction confusion matrix.

	Faulty	Non- Faulty
Faulty Modules	No. of True Positive (n(TP))	No. of False Negative (n(FN))
Non-Faulty Modules	No. of False Positive (n(FP))	No. of True Negative (n(TN))

confusion matrix in Table 2. n(TP), n(FN), n(FP), and n(TN) are the number of true faulty modules, the number false non-faulty modules, the number of false faulty modules and the number of true non-faulty modules respectively. Sensitivity is the percentage of actual faulty modules which is correctly identified, and Specificity is the percentage of non-faulty modules which is correctly identified. It is calculated by using below formula.

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{FP + TN}$$

ROC curve shows that the tradeoff between sensitivity and specificity. ROC curve is plotted with the false positive rate at x-axis and the true positive rate at y-axis. We utilize AUC evaluation measures to verify the performance on different compared methods. In machine learning, AUC is widely used to check the performance evaluation spatially for imbalanced classes.

D. EXPERIMENTAL DESIGN

To investigate the validity of our approach, we designed experiments to answer the research questions, five different schemes are designed accordingly to different combinations of feature extraction techniques with feature selection technique (RUS). These schemes are applied to three different prediction models to comprehensively study the effects on the performance of software fault prediction. In order to train our prediction models, we use average results over the 10×10 folds cross-validation approach [57], [84]. Cross-validation means that the data is equally divided into ten parts (i.e. 90% training and 10% testing dataset). To further reduce the effect of randomness, 10-folds cross-validation approach is repeated 10 times (100 iterations in total), we take the average of 100 times same experiments as the final value of Sensitivity, Specificity, and AUC to validate our results.

1) FEATURE EXTRACTION TECHNIQUE

We use two different feature extraction techniques in our experiments for the empirical study. One is our proposed DFCM clustering based feature extraction algorithm and another one is Information Gain (IG) [56], which only uses feature ranking. The further details of these two algorithms are described as follows.

a: INFORMATION GAIN (IG)

Information Gain is the probability-based measuring technique [39]. IG is an Entropy-Based technique, which measures the reduction in uncertainty of a class label after observing a feature. IG can be calculated by using following equation.

$$IG(X/Y) = h(X) - h(X/Y) \tag{2}$$

In equation 2, $h(X)$ is entropy of a discrete random variable X (i.e. the class) which is calculated by

$$h(X) = -\sum_{x \in X} p(x) \log_2 p(x) \tag{3}$$

Where $p(x)$ is the probability of x. $h(X/Y)$ is the conditional entropy, which calculates the uncertainty of X given the observed variable Y (i.e. the feature) which is calculated by

$$h(X/Y) = -\sum_{y \in Y} p(y) \sum_{x \in X} p(x/y) \log_2 p(x/y) \tag{4}$$

b: DFCM CLUSTERING BASED FEATURE EXTRACTION

As we discussed before, this algorithm is for creating new features by using the multi-clustering technique. There are two steps for feature extraction, first is the multi-clustering by using Fuzzy C-Mean rule [85]. Initially, the dataset converts into the binary cluster of labeled and unlabeled dataset, then these cluster lead to intra-cluster of classes and features. Total k(2+1) clusters are created, where k is the number of features. Calculate membership and centroid by using FCM on k(2+1) clusters. In the second step, feature mapping by activation function by using equation 1, which is non-linear mapping that attempts to be 'softer' while also keeping some sparsely. In this activation function, output 'o' is for any feature 'k', where the distance to the centroid is above average. From the results of the experiment in table 3, we can conclude that by this activation function almost half of the original features are selected except on some datasets.

2) FEATURE SELECTION TECHNIQUE

In this stage, we use random under-sampling (RUS) to build a more balanced feature by selecting 's' features, where 's' is the minimum number of features extracted in any clusters, which is shown in table3.

TABLE 3. Details of selected features by feature extraction algorithm with (RUS).

Dataset	No of feature extraction in all subsets (True, False, Unlabeled)	s (No of selected features by RUS)	Percentage of selected features by RUS
Eclipse 2.0	86, 81, 97	81	0.52
Eclipse 2.1	85, 86, 92	85	0.55
Eclipse 3.0	96, 86, 110	86	0.55
cm1	18, 19, 21	18	0.49
jm1	12, 15, 12	12	0.57
kc1	11, 9, 10	9	0.45
kc3	18, 20, 22	18	0.46
mc2	21, 21, 24	21	0.53
mw1	20, 19, 22	19	0.51
pc1	17, 16, 19	16	0.43
pc3	20, 22, 20	20	0.54
pc4	17, 18, 20	17	0.46
pc5	18, 19, 19	18	0.47
Avg			0.50

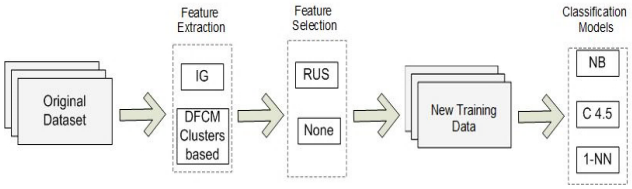

FIGURE 4. The configuration of empirical study.

Based on the two features extraction techniques and RUS, we can have five possible combinations to address the research question. First, the original dataset without any data pre-processing and it is donated by 'None'. Second and third, we use feature extraction techniques and denoted by 'IG' and 'DFCM' respectively. Fourth and fifth, we use RUS with feature extraction technique and denoted by 'IG+RUS' and 'DFCM+RUS' respectively. These all are shown in figure 4.

3) CLASSIFICATION MODELS

Our experiment is conducted on three widely used classification models in software fault prediction [Naive Bayes (NB) [13], C4.5 decision tree (C4.5) [86] and 1-nearest neighbor rule (1-NN) [77]. C4.5 is a decision tree algorithm, and information gain ratio is used for feature selection, which eliminates the bias of selecting frequent features of information gain (IG). Naive Bayes (NB) is a probabilistic classifier based on Bayes theorem and supposes that all the features are independent. 1-NN is an instance-based algorithm, and one sample can be classified by a majority role of its 1-nearest neighbor. All these three algorithms are implemented based on WEKA 3.5.5 [87] to avoid external threats to validity.

E. EXPERIMENTAL RESULT AND ANALYSIS

In this section, Table 4, 5, and 6 show the sensitivity and specificity results of all different data pre-processing schemes on three prediction models (NB, C4.5, and 1-NN) with labeled rate (10%, 20%, and 30%). Sensitivity measure the ration of actual faulty modules which are correctly identified and specificity measure the ratio of non-faulty modules which are correctly identified. Ideally, the results of both being high for better performance. According to Table 4, 5, and 6 we can analyze that our proposed data pre-processing scheme (DFCM+RUS) archives better sensitivity and specificity results in all prediction models with all labeled rates. Especially, our proposed schemes perform better on Naive Bayes prediction model. When we compared between sensitivity and specificity, sensitivity is better than specificity with all labeled rates.

Figure 5 (a), (b), and (c) are the comparison of sensitivity and specificity of proposed schemes with other data pre-processing schemes for three prediction models (NB, C4.5, & 1-NN) respectively. The results are the average of sensitivity or specificity on all labeled rates. Accordingly figure 5 (a), (b), and (c), our proposed schemes achieve better sensitivity and specificity on all prediction models.

Other than sensitivity and specificity, Table 7, 8, and 9 shows the AUC results after different schemes on three fault prediction models (NB, C4.5, and 1-NN respectively). Every Table has the list of five different schemes (None, IG, DFCM, IG+RUS, and DFCM+RUS). Every result shows the average of AUC across the 10×10 fold cross-validation on thirteen datasets with three different rates of labeled data. Bold values are the best performances in each row of the tables with different labeled rates.

The comparison of ROC curves of our proposed data pre-processing approach on three prediction model (NB, C4.5, and 1-NN) for three datasets (cm1, pc3, and Eclipse 2.0) are shown in figure 6. All ROC curves at labeled rate 0.3 and ROC curve shows the tradeoff between sensitivity and specificity. According to the ROC curves, our proposed approach (DFCM+RUS) achieve good AUC on Naïve Bayes prediction model. Another hand 1-NN prediction model performs better than C4.5 prediction model on our proposed approach.

To further analysis on the effectiveness of different schemes, we perform the Mann-Whitney U test (also called Wilcoxon rank-sum test) which is non-parametric rank-order

TABLE 4. Mean sensitivity and specificity of Naive Bayes on different data pre-processing schemes.

Dataset		Labeled Rate = 0.1					Labeled Rate = 0.2					Labeled Rate = 0.3				
		None	IG	DFCM	IG + RUS	DFCM + RUS	None	IG	DFCM	IG + RUS	DFCM + RUS	None	IG	DFCM	IG + RUS	DFCM + RUS
Eclipse 2.0	Sensitivity	0.80	0.81	0.79	0.80	**0.82**	0.81	0.83	0.83	0.80	**0.84**	0.80	0.81	0.80	0.78	**0.94**
	Specificity	0.77	0.79	0.73	0.78	**0.81**	**0.80**	0.77	0.80	0.73	**0.80**	0.74	0.79	0.80	0.72	**0.92**
Eclipse 2.1	Sensitivity	0.76	0.69	0.76	0.70	**0.84**	0.76	0.76	0.75	0.74	**0.83**	0.75	0.79	0.87	0.75	**0.89**
	Specificity	0.71	**0.78**	0.73	0.72	0.71	0.67	0.71	0.69	0.71	**0.79**	0.69	0.71	0.84	0.72	**0.86**
Eclipse 3.0	Sensitivity	0.77	0.79	0.79	0.76	**0.80**	0.76	0.76	0.81	0.80	**0.86**	0.76	0.77	0.80	0.75	**0.86**
	Specificity	0.78	0.78	**0.79**	0.74	0.78	0.78	0.81	0.77	0.77	**0.84**	0.78	0.79	0.76	0.74	**0.83**
cm1	Sensitivity	**0.79**	**0.79**	0.78	0.78	**0.79**	0.75	0.74	0.76	0.78	**0.79**	0.75	0.75	0.82	0.79	**0.83**
	Specificity	0.74	0.75	0.78	0.79	**0.80**	0.74	0.78	0.78	0.75	**0.81**	0.73	0.73	0.77	0.78	**0.82**
jm1	Sensitivity	0.72	0.73	0.75	0.71	**0.81**	0.71	0.79	0.83	0.74	**0.83**	0.71	0.72	0.75	0.80	**0.85**
	Specificity	0.69	0.76	0.79	**0.80**	0.78	0.78	0.78	**0.80**	0.75	0.76	0.79	0.79	**0.80**	0.76	**0.80**
kc1	Sensitivity	0.79	0.73	0.80	0.76	**0.83**	0.75	0.81	0.85	0.83	**0.87**	0.80	0.79	0.89	0.89	**0.91**
	Specificity	0.80	**0.81**	0.80	0.75	**0.81**	0.69	0.76	0.75	0.78	**0.77**	0.76	0.77	0.84	0.85	**0.90**
kc3	Sensitivity	0.73	0.77	0.85	0.74	**0.89**	0.80	0.79	0.84	0.81	**0.89**	0.83	0.80	0.89	0.90	**0.93**
	Specificity	0.80	0.76	0.80	0.76	**0.83**	0.78	0.77	0.81	0.80	**0.84**	0.79	0.75	0.81	0.81	**0.90**
mc2	Sensitivity	0.76	0.79	0.92	0.84	**0.94**	0.71	0.76	**0.95**	0.78	**0.95**	0.80	0.73	0.84	0.72	**0.97**
	Specificity	0.70	0.78	0.90	0.79	**0.92**	0.73	0.74	0.90	0.79	**0.92**	0.78	0.70	0.76	0.78	**0.94**
mw1	Sensitivity	0.71	0.73	0.77	0.77	**0.79**	0.71	0.80	0.76	0.80	**0.80**	0.73	0.79	0.81	0.78	**0.84**
	Specificity	0.75	**0.78**	0.76	**0.78**	0.71	0.73	0.75	**0.79**	0.75	0.75	0.70	0.75	0.77	0.78	**0.79**
pc1	Sensitivity	0.76	0.76	0.77	0.75	**0.83**	0.76	0.69	0.81	0.79	**0.86**	0.77	0.68	0.84	0.81	**0.87**
	Specificity	0.71	0.72	0.75	0.69	**0.80**	0.74	0.67	0.79	0.75	**0.83**	0.73	0.69	0.78	0.80	**0.85**
pc3	Sensitivity	0.75	0.75	0.78	0.79	**0.80**	0.76	0.78	0.81	**0.81**	0.80	0.79	0.76	0.86	0.82	**0.87**
	Specificity	0.73	0.74	0.77	0.77	**0.78**	0.75	0.76	0.78	0.77	**0.81**	0.78	0.79	0.81	0.81	**0.84**
pc4	Sensitivity	0.82	0.83	0.87	0.87	**0.89**	0.81	0.81	0.90	0.89	**0.93**	0.83	0.85	0.88	0.91	**0.96**
	Specificity	0.79	0.79	0.82	0.79	**0.85**	0.81	0.79	0.86	0.87	**0.90**	0.79	0.80	0.84	**0.86**	0.84
pc5	Sensitivity	0.89	0.90	0.96	0.90	**0.97**	0.89	0.90	0.93	0.81	**0.97**	0.90	0.92	0.94	0.95	**0.98**
	Specificity	0.86	0.85	0.84	0.89	**0.94**	0.87	0.87	0.92	0.86	**0.95**	0.85	0.90	0.89	0.92	**0.97**
Avg	Sensitivity	0.77	0.77	0.81	0.78	**0.85**	0.77	0.79	0.83	0.80	**0.86**	0.79	0.78	0.85	0.82	**0.90**
	Specificity	0.76	0.78	0.79	0.77	**0.81**	0.76	0.77	0.80	0.78	**0.83**	0.76	0.77	0.81	0.79	**0.87**

TABLE 5. Mean sensitivity and specificity of C4.5 on different data pre-processing schemes.

Dataset		Labeled Rate = 0.1					Labeled Rate = 0.2					Labeled Rate = 0.3				
		None	IG	DFCM	IG + RUS	DFCM + RUS	None	IG	DFCM	IG + RUS	DFCM + RUS	None	IG	DFCM	IG + RUS	DFCM + RUS
Eclipse 2.0	Sensitivity	0.64	0.69	0.72	0.70	**0.74**	0.65	0.71	0.75	0.74	**0.80**	0.65	0.70	0.73	0.70	**0.79**
	Specificity	**0.73**	0.71	0.71	0.67	0.71	0.69	0.69	0.69	0.70	**0.78**	0.68	0.71	0.70	0.69	**0.73**
Eclipse 2.1	Sensitivity	0.59	0.62	0.65	0.68	**0.74**	0.59	0.61	0.71	0.73	**0.78**	0.60	0.63	0.73	0.74	**0.78**
	Specificity	0.57	0.59	0.64	0.67	**0.74**	0.60	0.59	0.70	0.70	**0.73**	0.58	0.60	0.70	0.70	**0.74**
Eclipse 3.0	Sensitivity	0.59	0.63	0.72	0.69	**0.78**	0.65	0.70	0.70	0.72	**0.85**	0.62	0.72	0.72	0.73	**0.85**
	Specificity	0.57	0.61	0.69	0.70	**0.75**	0.62	0.65	0.69	0.67	**0.81**	0.61	0.70	0.71	0.71	**0.81**
cm1	Sensitivity	0.50	0.53	0.63	0.57	**0.65**	0.50	0.53	0.64	0.58	**0.69**	0.51	0.54	**0.64**	0.62	**0.64**
	Specificity	0.53	0.54	0.62	0.60	**0.67**	0.55	0.55	0.62	0.61	**0.67**	0.52	0.58	0.60	0.60	**0.75**
jm1	Sensitivity	0.55	0.68	0.68	0.67	**0.70**	0.57	0.65	0.65	0.65	**0.70**	0.57	0.65	0.65	0.66	**0.73**
	Specificity	0.51	0.59	0.62	0.65	**0.68**	0.54	0.61	0.64	0.64	**0.69**	0.55	0.62	0.64	0.62	**0.69**
kc1	Sensitivity	0.56	0.58	0.69	0.75	**0.77**	0.56	0.63	0.74	0.75	**0.77**	0.61	0.63	0.72	0.77	**0.79**
	Specificity	0.49	0.55	0.68	0.68	**0.71**	0.55	0.59	0.65	0.70	**0.72**	0.58	0.60	0.69	0.72	**0.75**
kc3	Sensitivity	0.58	0.64	0.65	0.65	**0.70**	0.63	0.67	0.70	0.66	**0.78**	0.63	0.69	0.69	0.70	**0.79**
	Specificity	0.55	0.61	0.64	0.62	**0.65**	0.59	0.61	0.65	0.65	**0.70**	0.57	0.65	0.67	0.68	**0.76**
mc2	Sensitivity	0.58	0.62	0.71	0.69	**0.89**	0.55	0.64	0.74	0.69	**0.90**	0.58	0.67	0.70	0.73	**0.93**
	Specificity	0.55	0.59	0.65	0.70	**0.85**	0.54	0.60	0.68	0.65	**0.86**	0.55	0.65	0.69	0.70	**0.89**
mw1	Sensitivity	0.50	0.55	0.63	0.56	**0.67**	0.51	0.58	0.65	0.62	**0.89**	0.48	0.63	0.64	0.62	**0.68**
	Specificity	0.46	0.60	0.58	0.54	**0.61**	0.47	0.52	0.60	0.57	**0.83**	0.39	0.59	0.62	0.62	**0.67**
pc1	Sensitivity	0.63	0.66	0.65	0.69	**0.67**	0.62	0.69	0.68	0.65	**0.76**	0.64	0.69	0.68	0.70	**0.77**
	Specificity	0.70	0.69	0.68	0.68	**0.70**	0.65	0.68	0.70	0.69	**0.73**	0.60	0.63	0.67	0.65	**0.75**
pc3	Sensitivity	0.60	0.61	0.65	0.65	**0.75**	0.57	0.61	0.67	0.69	**0.79**	0.61	0.61	0.67	0.71	**0.81**
	Specificity	0.56	0.56	0.60	0.64	**0.69**	0.58	0.58	0.60	0.67	**0.72**	0.54	0.59	0.60	0.67	**0.77**
pc4	Sensitivity	0.77	0.82	0.81	0.83	**0.85**	0.77	0.84	0.84	0.82	**0.86**	0.79	0.80	0.85	0.83	**0.87**
	Specificity	0.63	0.76	0.76	0.72	**0.80**	0.69	0.79	0.77	0.76	**0.81**	0.72	0.78	0.78	**0.80**	0.79
pc5	Sensitivity	0.77	0.86	0.86	0.90	**0.89**	0.80	0.89	0.89	0.90	**0.94**	0.80	0.90	0.89	0.91	**0.94**
	Specificity	0.73	0.80	0.83	**0.86**	0.84	0.76	0.80	0.75	**0.87**	**0.87**	0.75	0.85	0.86	0.86	**0.91**
Avg	Sensitivity	0.60	0.65	0.70	0.69	**0.75**	0.61	0.67	0.72	0.71	**0.81**	0.62	0.68	0.72	0.72	**0.80**
	Specificity	0.58	0.63	0.67	0.67	**0.72**	0.60	0.64	0.67	0.68	**0.76**	0.59	0.66	0.69	0.69	**0.77**

statistic test used to access whether two independent groups are significantly different from each other. For the entire test, the null hypothesis is that there is no difference between the two schemes, and the significance level of alpha (α) is 0.05

Table 7, 8, and 9 shows the Mann-Whitney U test results for the comparison of different schemes in terms of the average of AUC on all labeled rates. To perform the Mann-Whitney U test first ranks all the values from low to high. The smallest

TABLE 6. Mean sensitivity and specificity of 1-NN on different data pre-processing schemes.

Dataset		Labeled Rate = 0.1					Labeled Rate = 0.2					Labeled Rate = 0.3				
		None	IG	DFCM	IG + RUS	DFCM + RUS	None	IG	DFCM	IG + RUS	DFCM + RUS	None	IG	DFCM	IG + RUS	DFCM + RUS
Eclipse 2.0	Sensitivity	0.69	0.71	0.70	0.73	**0.84**	0.69	0.73	0.73	0.77	**0.80**	0.69	0.73	0.73	0.77	**0.79**
	Specificity	0.65	0.66	0.69	0.68	**0.79**	0.65	0.70	0.70	0.73	**0.77**	0.65	0.70	0.71	0.75	**0.74**
Eclipse 2.1	Sensitivity	0.68	0.69	0.72	0.70	**0.77**	0.67	0.67	0.74	0.72	**0.75**	0.68	0.68	0.67	0.74	**0.82**
	Specificity	0.64	0.65	0.70	0.67	**0.71**	0.68	0.65	0.70	**0.70**	**0.70**	0.67	0.67	0.70	0.70	**0.80**
Eclipse 3.0	Sensitivity	0.69	0.69	0.78	0.69	**0.79**	0.70	0.73	0.77	0.74	**0.85**	0.70	0.75	0.77	0.77	**0.84**
	Specificity	0.67	0.65	0.70	0.73	**0.75**	0.65	0.70	0.72	0.71	**0.80**	0.65	0.69	0.75	0.76	**0.80**
cm1	Sensitivity	0.58	0.60	0.63	0.65	**0.72**	0.58	0.59	0.69	0.64	**0.71**	0.60	0.62	0.65	0.65	**0.69**
	Specificity	0.60	0.61	0.65	0.65	**0.70**	0.65	0.64	0.67	0.69	**0.75**	0.69	0.63	0.64	0.64	**0.78**
jm1	Sensitivity	0.60	0.64	0.67	0.64	**0.73**	0.60	0.66	0.66	0.71	**0.77**	0.60	0.64	0.67	0.67	**0.80**
	Specificity	0.67	0.63	0.65	0.63	**0.68**	0.69	0.63	0.65	0.68	**0.73**	0.65	0.67	0.63	0.66	**0.75**
kc1	Sensitivity	0.65	0.69	0.74	0.74	**0.80**	0.65	0.70	0.75	0.73	**0.81**	0.65	0.69	0.75	0.71	**0.84**
	Specificity	0.60	0.65	0.68	0.71	**0.76**	0.62	0.65	0.72	0.70	**0.77**	0.62	0.66	0.70	0.70	**0.82**
kc3	Sensitivity	0.66	0.67	0.76	0.74	**0.79**	0.67	0.70	0.74	0.73	**0.85**	0.64	0.69	0.74	0.74	**0.85**
	Specificity	0.62	0.65	0.69	0.69	**0.74**	0.64	0.65	0.69	0.67	**0.82**	0.63	0.70	0.70	0.69	**0.78**
mc2	Sensitivity	0.59	0.64	0.87	0.80	**0.95**	0.59	0.63	0.87	0.90	**0.95**	0.64	0.61	0.90	0.91	**0.96**
	Specificity	0.58	0.61	0.84	0.76	**0.92**	0.60	0.59	0.86	0.85	**0.90**	0.58	0.57	0.84	0.86	**0.93**
mw1	Sensitivity	0.62	0.62	0.63	0.65	**0.67**	0.61	0.64	0.66	0.63	**0.72**	0.67	0.64	0.65	0.67	**0.78**
	Specificity	0.60	0.61	0.63	0.59	**0.64**	0.59	0.62	0.60	0.63	**0.70**	0.58	0.60	0.60	0.65	**0.73**
pc1	Sensitivity	0.70	0.72	0.76	0.76	**0.77**	0.68	0.75	0.76	0.72	**0.84**	0.70	0.72	0.75	0.75	**0.86**
	Specificity	0.65	0.70	0.68	0.72	**0.73**	0.69	0.73	0.71	0.68	**0.75**	0.68	0.69	0.71	0.73	**0.79**
pc3	Sensitivity	0.67	0.71	0.71	0.70	**0.75**	0.69	0.77	0.76	0.71	**0.80**	0.67	0.73	0.71	0.74	**0.87**
	Specificity	0.64	0.69	0.70	0.69	**0.73**	0.63	0.70	0.70	0.69	**0.75**	0.63	0.68	0.67	0.70	**0.84**
pc4	Sensitivity	0.74	0.83	0.81	0.81	**0.86**	0.74	0.85	0.88	0.84	**0.89**	0.75	0.84	0.75	0.85	**0.89**
	Specificity	0.69	0.78	**0.80**	0.74	**0.80**	0.69	0.81	0.79	0.79	**0.84**	0.69	**0.81**	0.80	0.80	**0.81**
pc5	Sensitivity	0.79	0.81	0.85	0.81	**0.93**	0.83	0.83	**0.86**	0.82	0.84	0.82	0.84	0.86	0.86	**0.87**
	Specificity	0.76	0.79	0.80	0.80	**0.87**	0.79	0.81	**0.84**	0.79	0.78	0.78	0.82	**0.84**	0.84	0.79
Avg	Sensitivity	0.67	0.69	0.74	0.72	**0.80**	0.67	0.71	0.76	0.74	**0.81**	0.68	0.71	0.74	0.76	**0.84**
	Specificity	0.64	0.67	0.71	0.70	**0.76**	0.66	0.68	0.72	0.72	**0.77**	0.65	0.68	0.71	0.73	**0.80**

number gets a rank of 1 and the largest number gets a rank of ''n'', where ''n'' is the total number of values in two groups. U-Value and P-Value are the Mann-Whitney U-Value and Mann-Whitney P-Value. If $\boldsymbol{P} < \mathbf{0.05}$, it means the comparison is significantly different which is in bold font.

RQ1. In our feature extraction stage, whether FCM based semi-supervised multi-clustering pre-processing data can improve the performance of the classification models with imbalanced classes?

To investigate the effectiveness of our approach on fault prediction models, we will analyze the results on two factors first is any pre-processing date approach can affect the prediction performance. Second, our proposed approach achieves how much success for fault prediction performance from Table 7, 8, and 9, We can analysis, two pre-processing approaches (IG+RUS and DFCM+RUS) on thirteen dataset perform better than AUC results on all prediction models (NB, C4.5, and 1-NN) with different labeled rates.

If we will analysis according to the prediction models than by using Naive Bayes the preprocessing data approach (IG+RUS) can achieve 1.5%, 3.5%, & 4.3% improve AUC measure with (0.1, 0.2, and 0.3) labeled rates and our proposed approach (DFCM+*RUS*) can achieve 6.7%, 8.7%, & 11.5% improve AUC measure with all labeled rates. Similarly, for the other two prediction models (C4.5, and 1-NN), the pre-processing data approach IG+RUS can achieve improve AUC measure 8.7%, 89%, & 10.1% on C4.5 and 5.3%, 7%, & 7.9% on 1-NN with all labeled rates.

Our proposed approach DFCM+RUS can achieve improve AUC measure by 13.9%, 18.1% & 18.4% on C4.5 and 11.4%, 13.3%, & 15.5% on 1-NN will all labeled rates. We can analysis by the above result the preprocessing data approach can improve the prediction performance. On the bases of average the IG+RUS approach achieve 31%, 9.23%, & 6.73% improve AUC measure on NB, C4.5, and 1-NN and our proposed approach DFCM+RUS can achieve 8.97%, 16.8%, & 13.4% improve AUC measure o NB, C4.5, and 1-NN. C4.5 can achieve the highest improvement in AUC measure compared to NB and 1-NN. We will also do the comparison of prediction performance of fault prediction models with and without pre-processing data approaches by Mann-Whitney U test results in Table 10. The P-Values of Mann-Whitney U test (0.03572, 0.00018, 0.00880, 0.00034, 0.00758, and 0.00014) are less than 0.05, which shows that the pre-processing data approach significantly improve prediction models no matter which prediction model is used.

RQ2. Does combining the feature selection technique with our DFCM based feature extraction technique together for pre-processing data have more significantly improved than by using any of the individual feature selection for prediction performance?

For the integration of this research question, we will analyze the results of prediction models by comparison of two schemes, DFCM vs IG with and without RUS. From AUC Table 7, 8, and 9, the results of mean AUC on thirteen dataset of NASA & Eclipse, DFCM feature extraction approach can adore improve AUC measure on IG approach by 5.2%, 5.2%,

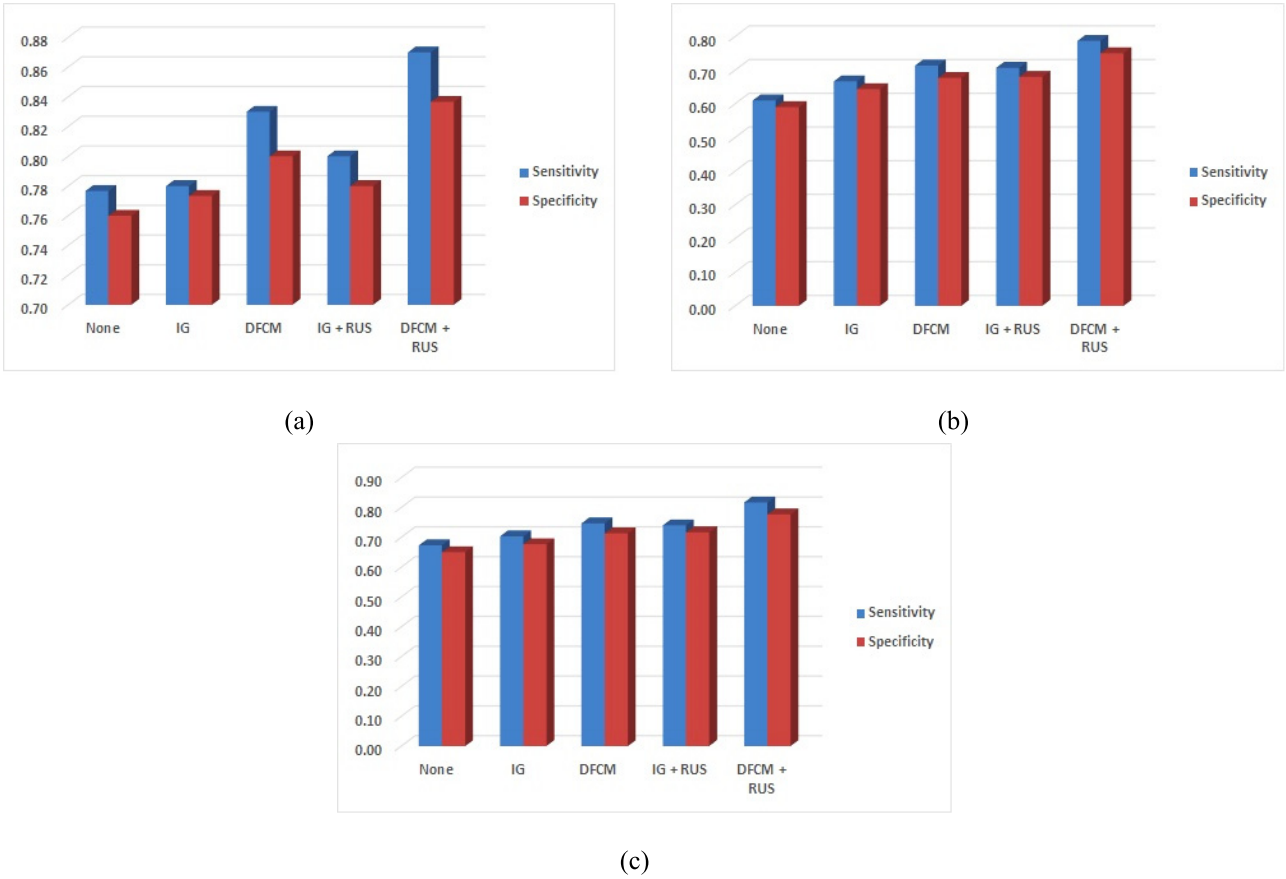

FIGURE 5. Comparison of sensitivity and specificity of proposed schemes with other data pre-processing schemes for three prediction models (NB, C4.5, & 1-NN). (a) NB. (b) C4.5. (c) 1-NN.

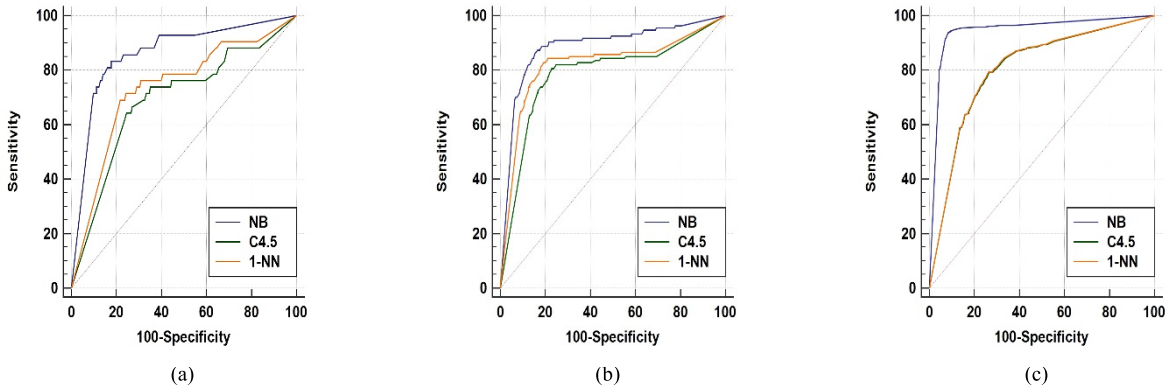

FIGURE 6. Comparison of ROC curve of proposed schemes on three prediction model (NB, C4.5, and 1-NN) for three datasets (cm1, pc3, and Eclipse 2.0) with labeled rate 0.3. (a) cm1. (b) pc3. (c) Eclipse 2.0.

& 5.2%, with RUS and 4.5%, 5.4%, & 6% without RUS on Naive Bayes prediction model. Similarly DFCM approach can achieve improve AUC measure on IG approach by 5.2%, 8.2%, & 8.3%, and 6.1%, 6.3%, & 7.6% with RUS and 4.8%, 3.1%, & 3.7 %, and 5.1%, 4.5%, & 4.7% without RUS on C4.5 and 1-NN prediction models respectively.

From the above results, our proposed approach achieves good AUC measure on all prediction models. Table 11, shows the summarized results of Mann-Whitney U test on DFCM vs IG on three prediction models (NB, C4.5, and 1-NN) with and without RUS. We can conclude the result of Mann-Whitney U test of AUC, DFCM shows better results on all prediction models except C4.5. Average twelve out of thirteen dataset have improved their prediction performance. All P-Values of Mann-Whitney U test is less than 0.05, except for C4.5 predictive model. We can also analysis from the Table 11, the combination of Random Under-Sampling (RUS) with our approach DFCM boost the

TABLE 7. Mean AUC of Naive Bayes on different data pre-processing schemes.

Dataset	Labeled Rate = 0.1					Labeled Rate = 0.2					Labeled Rate = 0.3				
	None	IG	DFCM	IG + RUS	DFCM + RUS	None	IG	DFCM	IG + RUS	DFCM + RUS	None	IG	DFCM	IG + RUS	DFCM + RUS
Eclipse 2.0	0.783	0.790	0.801	0.798	**0.819**	0.792	0.798	0.813	0.802	**0.822**	0.802	0.791	0.835	0.800	**0.945**
Eclipse 2.1	0.791	0.735	0.759	0.733	**0.772**	0.737	0.774	0.759	0.759	**0.807**	0.753	0.781	0.889	0.758	**0.913**
Eclipse 3.0	0.766	0.768	0.799	0.767	**0.803**	0.759	0.793	0.803	0.797	**0.888**	0.760	0.795	0.824	0.805	**0.883**
cm1	0.752	0.749	0.803	0.752	**0.837**	0.763	0.751	0.833	0.790	**0.840**	0.768	0.761	0.798	0.831	**0.862**
jm1	0.766	0.769	0.799	0.770	**0.809**	0.776	0.769	0.801	0.769	**0.809**	0.776	0.774	0.813	0.796	**0.837**
kc1	0.767	0.783	0.837	0.782	**0.852**	0.772	0.804	0.838	0.828	**0.858**	0.791	0.807	0.852	0.847	**0.899**
kc3	0.772	0.761	0.849	0.773	**0.866**	0.780	0.774	0.853	0.834	**0.872**	0.812	0.793	0.880	0.884	**0.906**
mc2	0.721	0.808	0.911	0.811	**0.928**	0.721	0.750	0.937	0.767	**0.947**	0.783	0.725	0.823	0.778	**0.966**
mw1	0.733	0.745	0.760	**0.762**	0.760	0.729	0.783	0.788	0.781	**0.797**	0.727	0.788	0.798	0.794	**0.822**
pc1	0.742	0.743	0.800	0.732	**0.822**	0.743	0.691	0.828	0.770	**0.847**	0.764	0.691	0.821	0.827	**0.889**
pc3	0.770	0.773	0.790	0.793	**0.799**	0.779	0.790	0.807	0.806	**0.826**	0.789	0.793	0.859	0.836	**0.885**
pc4	0.806	0.816	0.878	0.856	**0.888**	0.820	0.831	0.891	0.882	**0.915**	0.823	0.833	0.887	0.897	**0.955**
pc5	0.881	0.896	0.943	0.910	**0.967**	0.888	0.899	0.950	0.931	**0.963**	0.890	0.902	0.933	0.946	**0.979**
Avg	0.773	0.780	0.825	0.788	**0.840**	0.774	0.785	0.839	0.809	**0.861**	0.788	0.787	0.847	0.831	**0.903**

TABLE 8. Mean AUC of C4.5 on different data pre-processing schemes.

Dataset	Labeled Rate = 0.1					Labeled Rate = 0.2					Labeled Rate = 0.3				
	None	IG	DFCM	IG + RUS	DFCM + RUS	None	IG	DFCM	IG + RUS	DFCM + RUS	None	IG	DFCM	IG + RUS	DFCM + RUS
Eclipse 2.0	0.661	0.701	0.706	0.704	**0.713**	0.664	0.703	0.710	0.708	**0.790**	0.667	0.704	0.723	0.709	**0.803**
Eclipse 2.1	0.584	0.618	0.699	0.692	**0.731**	0.586	0.609	0.701	0.713	**0.760**	0.588	0.619	0.716	0.724	**0.805**
Eclipse 3.0	0.634	0.694	0.710	0.706	**0.769**	0.637	0.698	0.710	0.710	**0.843**	0.640	0.701	0.719	0.720	**0.843**
cm1	0.504	0.548	0.612	0.583	**0.673**	0.506	0.550	0.613	0.592	**0.680**	0.508	0.552	0.618	0.609	**0.692**
jm1	0.536	0.615	0.621	0.628	**0.663**	0.553	0.623	0.628	0.629	**0.692**	0.553	0.624	0.629	0.640	**0.704**
kc1	0.584	0.606	0.706	0.741	**0.761**	0.591	0.611	0.716	0.743	**0.750**	0.601	0.616	0.708	0.750	**0.783**
kc3	0.602	0.654	0.680	0.676	**0.724**	0.605	0.660	0.689	0.681	**0.764**	0.608	0.660	0.688	0.712	**0.789**
mc2	0.566	0.608	0.712	0.682	**0.879**	0.562	0.608	0.717	0.684	**0.895**	0.560	0.690	0.723	0.714	**0.913**
mw1	0.496	0.570	0.611	0.570	**0.654**	0.493	0.573	0.631	0.600	**0.884**	0.450	0.569	0.638	0.628	**0.694**
pc1	0.648	0.677	0.679	0.688	**0.691**	0.650	0.676	0.680	0.689	**0.749**	0.653	0.681	0.683	0.690	**0.788**
pc3	0.586	0.590	0.638	0.677	**0.733**	0.589	0.593	0.641	0.681	**0.751**	0.593	0.601	0.646	0.683	**0.785**
pc4	0.750	0.806	0.800	0.808	**0.842**	0.752	0.812	0.810	0.811	**0.843**	0.772	0.820	0.817	0.821	**0.855**
pc5	0.763	0.853	0.867	**0.889**	0.888	0.777	0.861	0.870	0.880	**0.923**	0.789	0.870	0.887	0.898	**0.924**
Avg	0.609	0.657	0.695	0.696	**0.748**	0.613	0.660	0.701	0.702	**0.794**	0.614	0.670	0.707	0.715	**0.798**

TABLE 9. Mean AUC of 1-NN on different data pre-processing schemes.

Dataset	Labeled Rate = 0.1					Labeled Rate = 0.2					Labeled Rate = 0.3				
	None	IG	DFCM	IG + RUS	DFCM + RUS	None	IG	DFCM	IG + RUS	DFCM + RUS	None	IG	DFCM	IG + RUS	DFCM + RUS
Eclipse 2.0	0.670	0.706	0.710	0.708	**0.812**	0.672	0.716	0.715	0.760	**0.784**	0.675	0.709	0.716	0.761	**0.805**
Eclipse 2.1	0.659	0.661	0.731	0.692	**0.749**	0.689	0.662	**0.736**	0.716	0.731	0.690	0.663	0.738	0.736	**0.845**
Eclipse 3.0	0.685	0.674	0.759	0.717	**0.766**	0.689	0.713	0.759	0.726	**0.842**	0.691	0.716	0.760	0.762	**0.843**
cm1	0.604	0.619	0.668	0.640	**0.712**	0.606	0.624	0.667	0.691	**0.730**	0.610	0.618	0.672	0.673	**0.736**
jm1	0.627	0.629	0.662	0.636	**0.714**	0.628	0.646	0.668	0.701	**0.761**	0.629	0.647	0.668	0.669	**0.783**
kc1	0.638	0.682	0.722	0.711	**0.781**	0.640	0.683	0.731	0.710	**0.791**	0.643	0.685	0.733	0.731	**0.826**
kc3	0.653	0.692	0.725	0.729	**0.768**	0.658	0.692	0.728	0.709	**0.832**	0.655	0.700	0.736	0.729	**0.841**
mc2	0.603	0.614	0.881	0.780	**0.926**	0.607	0.619	0.889	0.891	**0.929**	0.620	0.607	0.891	0.890	**0.949**
mw1	0.612	0.615	0.633	0.620	**0.648**	0.613	0.621	0.640	0.623	**0.713**	0.623	0.622	0.641	0.643	**0.723**
pc1	0.692	0.730	0.739	0.742	**0.752**	0.691	0.739	0.746	0.716	**0.802**	0.692	0.740	0.740	0.742	**0.844**
pc3	0.651	0.703	0.717	0.706	**0.743**	0.653	0.717	0.715	0.707	**0.783**	0.653	0.718	0.716	0.716	**0.821**
pc4	0.724	0.805	0.822	0.806	**0.830**	0.726	0.824	0.822	0.811	**0.858**	0.729	0.827	0.829	0.829	**0.869**
pc5	0.801	0.814	0.840	0.820	**0.906**	0.803	0.816	**0.847**	0.817	0.842	0.805	0.823	0.851	0.850	**0.845**
Avg	0.663	0.688	0.739	0.716	**0.777**	0.667	0.698	0.743	0.737	**0.800**	0.670	0.698	0.745	0.749	**0.825**

prediction performance. DFCM+RUS shows better results than IG+RUS on all prediction models. All P-Values of Mann-Whitney U test is less than 0.05, which shows that DFCM feature extraction approaches are significantly improved feature ranking approach (IG).

RQ3. Does combining RUS approach with DFCM together have a more significant improvement in prediction performance?

To investigate the prediction performance of DFCM feature extraction approach together with RUS, we will

TABLE 10. Mann-Whitney U test results for comparison between schemes with and without data pre-processing for classification models.

Classification Model	Scheme A vs. Scheme B	Sum of Rank		U-Value	Z-Value	P-Value	Significance
		Scheme A	Scheme B				
NB	IG + RUS Vs None	217	134	43	2.10256	**0.03572**	Significance
	DFCM + RUS Vs None	249	102	11	3.74359	**0.00018**	Significance
C4.5	IG + RUS Vs None	227	124	33	2.61538	**0.00880**	Significance
	DFCM + RUS Vs None	246	105	14	3.58974	**0.00034**	Significance
1-NN	IG + RUS Vs None	228	123	32	2.66607	**0.00758**	Significance
	DFCM + RUS Vs None	250.5	100.5	9.5	3.82051	**0.00014**	Significance

TABLE 11. Mann-Whitney U test results for comparison between schemes with and without data pre-processing for classification models.

Classification Model	Scheme A vs. Scheme B	Sum of Rank		U-Value	Z-Value	P-Value	Significance
		Scheme A	Scheme B				
NB	DFCM Vs IG	236	115	24	3.07692	**0.00208**	Significance
	DFCM+RUS Vs IG+RUS	234	117	26	2.97436	**0.00298**	Significance
C4.5	DFCM Vs IG	211	140	49	1.79487	0.07346	Non-Significance
	DFCM+RUS Vs IG+RUS	224	127	36	2.46154	**0.01390**	Significance
1-NN	DFCM Vs IG	215	136	45	2	**0.04550**	Significance
	DFCM+RUS Vs IG+RUS	220	131	40	2.25641	**0.02382**	Significance

TABLE 12. Mann-Whitney U test results for comparison between schemes with and without data pre-processing for classification models.

Classification Model	Scheme A vs. Scheme B	Sum of Rank		U-Value	Z-Value	P-Value	Significance
		Scheme A	Scheme B				
NB	IG+RUS Vs IG	208.5	142.5	51.5	1.66667	0.09492	Non-Significance
	DFCM+RUS Vs DFCM	223	128	37	2.41026	**0.01596**	Significance
C4.5	IG+RUS Vs IG	207	144	53	1.5897	0.11184	Non-Significance
	DFCM+RUS Vs DFCM	228.5	122.5	31.5	2.69231	**0.00714**	Significance
1-NN	IG+RUS Vs IG	210.5	140.5	49.5	1.76923	0.76720	Non-Significance
	DFCM+RUS Vs DFCM	215	136	45	2	**0.04550**	Significance

analyse the results of DFCM with and without on three fault prediction modules. AUC result of DFCM with and without RUS shown in Table 7, 8, and 9. From these results DFCM approach achieve improve AUC measure after RUS by 1.5%, 2.2%, & 5.5% on Naive Bayes, 5.3%, 9.3%, and 9.1% on C4.5 and 3.8%, 5.7%, & 8% on 1-NN.

From above results, we can conclude that RUS can improve the prediction performance after together with the DFCM approach. We can also conclude the results of Mann-Whitney U test in Table 12 by comparison of IG+RUS vs IG, and DFCM+RUS vs DFCM. According to the results, the combination of RUS with feature ranking approach (IG) is not significantly improved prediction performance. All P-Values of Mann-Whitney U test on IG+RUS vs IG with all predictive models is greater than 0.05. Another hand, the combination of RUS with feature extraction approach DFCM improve the prediction performance. All P-Values of Mann-Whitney U test on DFCM+RUS vs DFCM is less than 0.05, which shows that DFCM+RUS are significantly improved the DFCM approach.

V. THREATS TO VALIDITY

Some potential threats to validity right are affected by our experimental study.

A. EXTERNAL VALIDITY

Dataset quality may be the most important threat to the external validity. These threats refer to the generalizability of our experimental results. To ensure the representativeness of our study, we used NASA and Eclipse dataset which are commonly used for software fault prediction. In addition, we choose Fuzzy C-Mean clustering for the extension of multi-clustering and random under-sampling is used for balancing the dataset, which is widely used in software fault prediction, to ensure the soundness of our results.

B. INTERNAL VALIDITY

These threats refer to experimenter biases. To avoid this type of threat, all implementation is cross-checked by our research group. Withal, we perform our experiment 100 times and report the average performance over 100 runs and prediction models are provided by the commonly used WEKA package. Moreover, our experimental results are generated based on a large collection of the unbalanced dataset from public PROMISE repository, so the dataset is carefully examined whether the non-numeric feature is eliminated. Thus, we believe there are minimal threats to internal validity.

C. CONSTRUCT VALIDITY

These threats refer to the approximations of our evaluation measure. The initial threat to construct the validity of our work is that we assume that all the faults that we utilized in our study had some weights. We make utilization of sensitivity, specificity, and AUC to evaluate the software fault prediction. AUC has lower variance and is more reliable to indicate the predictive potential of the method when compared with another performance measures, such as precision, recall or

F-measure. Finally, we do the Mann-Whitney U test to further validate the significance of the differences in performance.

VI. CONCLUSION

In this paper, we provided an empirical study of semi-supervised DFCM based feature extraction data pre-processing approach, which incorporates the semi-supervised multi-clustering by using FCM clustering for Novel feature extraction technique with RUS to amend the quality of software dataset utilized by software fault prediction models with imbalanced classes. In this paper, we presented how ''Deep'' multi-cluster can be amalgamated for feature extraction technique and how it can avail to incorporate simultaneously unlabeled data and labeled data into the sub-clustering process and withal handle the class imbalance problem.

In our empirical study, we investigate the performance stability of our proposed data pre-processing approach on fault prediction models (NB, C4.5, & 1-NN). Experiment result demonstrates the potential of our approach in enhancing fault prediction performance on ten NASA dataset and three Eclipse dataset. The proposed method has the best AUC measure on all prediction models. But NB prediction model has achieved good AUC measure compared to C4.5 and 1-NN by using our proposed approach DFCM feature extraction with RUS. The rank test using Mann-Whitney U test, experiment results shows that the difference between DFCM+RUS scheme and our four schemes are statistically significant. After analysis of experiment results, our pre-processing data approach boosts the performance of accuracy on traditional prediction model with imbalance class ratio.

In our future work, we plan to extend our approach for multi-imbalance classes with the big dataset.

REFERENCES

[1] X.-Y. Jing, S. Ying, Z.-W. Zhang, S.-S. Wu, and J. Liu, ''Dictionary learning based software defect prediction,'' in *Proc. IEEE Int. Conf. Softw. Eng.*, May/Jun. 2014, pp. 414–423.

[2] S. Kim, H. Zhang, R. Wu, and L. Gong, ''Dealing with noise in defect prediction,'' in *Proc. IEEE Int. Conf. Softw. Eng.*, May 2011, pp. 481–490.

[3] H. Wang, T. M. Khoshgoftaar, and A. Napolitano, ''A comparative study of ensemble feature selection techniques for software defect prediction,'' in *Proc. Int. Conf. Mach. Learn. Appl.*, 2010, pp. 135–140.

[4] S. Wang and X. Yao, ''Using class imbalance learning for software defect prediction,'' *IEEE Trans. Rel.*, vol. 62, no. 2, pp. 434–443, Jun. 2013.

[5] T. Menzies, J. Greenwald, and A. Frank, ''Data mining static code attributes to learn defect predictors,'' *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 2–13, Jan. 2007.

[6] N. Nagappan and T. Ball, ''Static analysis tools as early indicators of pre-release defect density,'' in *Proc. Int. Conf. Softw. Eng.*, Saint Louis, MO, USA, May 2005, pp. 580–586.

[7] K. O. Elish and M. O. Elish, ''Predicting defect-prone software modules using support vector machines,'' *J. Syst. Softw.*, vol. 81, no. 5, pp. 649–660, 2008.

[8] B. Turhan and A. Bener, ''Analysis of Naive Bayes' assumptions on software fault data: An empirical study,'' *Data Knowl. Eng.*, vol. 68, no. 2, pp. 278–290, 2009.

[9] C. Catal and B. Diri, ''Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem,'' *Inf. Sci.*, vol. 179, no. 8, pp. 1040–1058, 2009.

[10] P. Singh, N. R. Pal, S. Verma, and O. P. Vyas, ''Fuzzy rule-based approach for software fault prediction,'' *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 5, pp. 826–837, May 2017.

[11] W. Liu, S. Liu, Q. Gu, J. Chen, X. Chen, and D. Chen, ''Empirical studies of a two-stage data preprocessing approach for software fault prediction,'' *IEEE Trans. Rel.*, vol. 65, no. 1, pp. 38–53, Mar. 2016.

[12] T. M. Khoshgoftaar and N. Seliya, ''Fault prediction modeling for software quality estimation: Comparing commonly used techniques,'' *Empirical Softw. Eng.*, vol. 8, pp. 255–283, Sep. 2003.

[13] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, ''Benchmarking classification models for software defect prediction: A proposed framework and novel findings,'' *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 485–496, Jul. 2008.

[14] K. P. Bennett and A. Demiriz, ''Semi-supervised support vector machines,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 11, 1999, pp. 368–374.

[15] T. Joachims, ''Transductive inference for text classification using support vector machines,'' in *Proc. Int. Conf. Mach. Learn.*, 1999, pp. 200–209.

[16] M. Belkin, P. Niyogi, and V. Sindhwani, '' Manifold regularization: A geometric framework for learning from labeled and unlabeled examples,'' *Mach. Learn. Res.*, vol. 7, pp. 2399–2434, Nov. 2006.

[17] X. Ao *et al.*, ''Combining supervised and unsupervised models via unconstrained probabilistic embedding,'' *Inf. Sci.*, vol. 257, pp. 101–114, Feb. 2014.

[18] S. Basu, A. Banerjee, and R. Mooney, ''Semi-supervised clustering by seeding,'' in *Proc. Int. Conf. Mach. Learn.*, 2002, pp. 27–34.

[19] W. Cai, S. Chen, and D. Zhang, ''A simultaneous learning framework for clustering and classification,'' *Pattern Recognit.*, vol. 42, no. 7, pp. 1248–1259, 2009.

[20] N. V. Chawla and G. J. Karakoulas, ''Learning from labeled and unlabeled data an empirical study across techniques and domains,'' *J. Artif. Intell. Res.*, vol. 23, pp. 331–366, Mar. 2005.

[21] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann, 1993.

[22] G. H. John and P. Langley, ''Estimating continuous distributions in Bayesian classifiers,'' in *Proc. 11th Conf. Uncertainty Artif. Intell.*, Montréal, QC, Canada, Aug. 1995, pp. 338–345.

[23] L. Breiman, ''Random forests,'' *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[24] M. D'Ambros, M. Lanza, and R. Robbes, ''An extensive comparison of bug prediction approaches,'' in *Proc. 7th IEEE Work. Conf. Mining Softw. Repositories (MSR)*, May 2010, pp. 31–41.

[25] M. Dash and H. Liu, ''Consistency-based search in feature selection,'' *Artif. Intell.*, vol. 151, nos. 1–2, pp. 155–176, 2003.

[26] J. Davis and M. Goadrich, ''The relationship between Precision-Recall and ROC curves,'' in *Proc. ACM 23rd Int. Conf. Mach. Learn.*, 2006, pp. 233–240.

[27] T. Fawcett, ''An introduction to ROC analysis,'' *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.

[28] N. E. Fenton and M. Neil, ''A critique of software defect prediction models,'' *IEEE Trans. Softw. Eng.*, vol. 25, no. 5, pp. 675–689, Sep./Oct. 1999.

[29] W. W. Cohen, ''Fast effective rule induction,'' in *Proc. Int. Conf. Mach. Learn.*, 1995, pp. 115–123.

[30] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 2012.

[31] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York, NY, USA: Wiley, 1973.

[32] S. Vluymans, D. S. Tarragó, Y. Saeys, C. Cornelis, and F. Herrera, ''Fuzzy rough classifiers for class imbalanced multi-instance data,'' *Pattern Recognit.*, vol. 53, pp. 36–45, May 2016.

[33] V. Wu *et al.*, ''Top 10 algorithms in data mining,'' *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008.

[34] R. Jensen and C. Cornelis, ''Fuzzy-rough nearest neighbor classification,'' in *Transactions on Rough Sets XIII*, J. Peters, A. Skowron, C. C. Chan, J. W. Grzymala-Busse, and W. P. Ziarko, Eds. Berlin, Germany: Springer, 2011, pp. 56–72.

[35] R. B. Bhatt and M. Gopal, ''FRCT: Fuzzy-rough classification trees,'' *Pattern Anal. Appl.*, vol. 11, no. 1, pp. 73–88, 2008.

[36] R. Jensen and C. Cornelis, ''Fuzzy-rough nearest neighbour classification and prediction,'' *Theor. Comput. Sci.*, vol. 412, no. 42, pp. 5871–5884, 2011.

[37] E. Ramentol *et al.*, ''IFROWANN: Imbalanced fuzzy-rough ordered weighted average nearest neighbor classification,'' *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 5, pp. 1622–1637, Oct. 2015.

[38] R. R. Yager, ''On ordered weighted averaging aggregation operators in multicriteria decisionmaking,'' *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. SMC-18, no. 1, pp. 183–190, Jan./Feb. 1988.

[39] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artif. Intell.*, vol. 97, nos. 1–2, pp. 245–271, 1997.
[40] H. Liu and H. Motoda, *Feature Extraction, Construction and Selection: A Data Mining Perspective*, vol. 453. London, U.K.: Springer, 2012.
[41] A. Arshad, S. Riaz, L. Jiao, and A. Murthy, "A semi-supervised deep fuzzy C-mean clustering for two classes classification," in *Proc. IEEE 3rd Inf. Technol. Mechatronics Eng. Conf. (ITOEC)*, Chongqing, China, Oct. 2017, pp. 365–370.
[42] A. Arshad, S. Riaz, L. Jiao, and A. Murthy, "Semi-supervised deep fuzzy c-mean clustering for software fault prediction," *IEEE Access*, vol. 6, pp. 25675–25685, 2018.
[43] G. Forestier and C. Wemmert, "Semi-supervised learning using multiple clusterings with limited labeled data," *Inf. Sci.*, vols. 361–362, pp. 48–65, Sep. 2016.
[44] N. R. Pal and V. K. Eluri, "Two efficient connectionist schemes for structure preserving dimensionality reduction," *IEEE Trans. Neural Netw.*, vol. 9, no. 6, pp. 1142–1154, Nov. 1998.
[45] N. R. Pal, V. K. Eluri, and G. K. Mandal, "Fuzzy logic approaches to structure preserving dimensionality reduction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 3, pp. 277–286, Jun. 2002.
[46] N. R. Pal and K. K. Chintalapudi, "A connectionist system for feature selection," *Neural Parallel Sci. Comput.*, vol. 5, no. 3, pp. 359–382, 1997.
[47] D. Chakraborty and N. R. Pal, "A neuro-fuzzy scheme for simultaneous feature selection and fuzzy rule-based classification," *IEEE Trans. Neural Netw.*, vol. 15, no. 1, pp. 110–123, Jan. 2004.
[48] R. O. Duda and P. Hart, *Pattern Classification and Scene Analysis*. New York, NY, USA: Wiley, 1973.
[49] S. Kim, E. J. Whitehead, Jr., and Y. Zhang, "Classifying software changes: Clean or buggy?" *IEEE Trans. Softw. Eng.*, vol. 34, no. 2, pp. 181–196, Mar./Apr. 2008.
[50] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," in *Proc. Int. Conf. Data Mining*, 2006, pp. 965–969.
[51] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[52] T. Jiang, L. Tan, and S. Kim, "Personalized defect prediction," in *Proc. Int. Conf. Autom. Softw. Eng.*, 2013, pp. 279–289.
[53] F. Zhang, A. Mockus, I. Keivanloo, and Y. Zou, "Towards building a universal defect prediction model," in *Proc. Work. Conf. Mining Softw. Repositories*, 2014, pp. 182–191.
[54] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Mach. Learn.*, vol. 39, nos. 2–3, pp. 103–134, 2000.
[55] H. Lu, B. Cukic, and M. Culp, "An iterative semi-supervised approach to software fault prediction," in *Proc. 7th Int. Conf. Predictive Models Softw. Eng.*, 2011, Art. no. 15.
[56] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, "Choosing software metrics for defect prediction: An investigation on feature selection techniques," *Softw.-Practice Exper.*, vol. 41, no. 5, pp. 579–606, 2011.
[57] S. Shivaji, E. J. Whitehead, R. Akella, and S. Kim, "Reducing features to improve code change-based bug prediction," *IEEE Trans. Softw. Eng.*, vol. 39, no. 4, pp. 552–569, Apr. 2013.
[58] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *J. Mach. Learn. Res.*, vol. 5, pp. 1205–1224, Oct. 2004.
[59] I. Jamali, M. Bazmara, and S. Jafari, "Feature selection in imbalance data sets," *Int. J. Comput. Sci. Issues*, vol. 9, no. 3, pp. 1–4, May 2012.
[60] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the NASA software defect datasets," *IEEE Trans. Softw. Eng.*, vol. 39, no. 9, pp. 1208–1215, Sep. 2013.
[61] K. Dejaeger, T. Verbraken, and B. Baesens, "Toward comprehensible software fault prediction models using Bayesian network classifiers," *IEEE Trans. Softw. Eng.*, vol. 39, no. 2, pp. 237–257, Feb. 2013.
[62] J. J. Peterson, "Regression analysis of count data," *Technometrics*, vol. 41, no. 4, p. 371, 1999, doi: 10.1080/00401706.1999.10485941.
[63] A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist. (AISTATS)*, vol. 15, 2011, pp. 215–223.
[64] T. M. Khoshgoftaar, C. Seiffert, J. Van Hulse, A. Napolitano, and A. Folleco, "Learning with limited minority class data," in *Proc. Int. Conf. Mach. Learn. Appl.*, 2007, pp. 348–353.
[65] H. Wang, T. M. Khoshgoftaar, and A. Napolitano, "An empirical study on the stability of feature selection for imbalanced software engineering data," in *Proc. 11th Int. Conf. Mach. Learn. Appl.*, Boca Raton, FL, USA, vol. 1, Dec. 2012, pp. 317–323.
[66] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intell. Data Anal.*, vol. 6, no. 5, pp. 429–449, Oct. 2002.
[67] T. G. Grbac, G. Mausa, and B. D. Basic, "Stability of software defect prediction in relation to levels of data imbalance," in *Proc. 2nd Workshops Softw. Qual. Anal., Monit., Improvement, Appl.*, Novi Sad, Serbia, Sep. 2013, pp. 1–10.
[68] D. Ryu, O. Choi, and J. Baik, "Value-cognitive boosting with a support vector machine for cross-project defect prediction," *Empirical Softw. Eng.*, vol. 21, no. 1, pp. 43–71, 2016.
[69] Q. Yu, S. Jiang, and Y. Zhang, "The performance stability of defect prediction models with class imbalance: An empirical study," *IEICE Trans. Inf. Syst.*, vol. E100-D, pp. 265–272, Feb. 2017.
[70] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, 1991.
[71] S. Le Cessie and J. C. Van Houwelingen, "Ridge estimators in logistic regression," *Appl. Statist.*, vol. 41, no. 1, pp. 191–201, 1992.
[72] J.-G. Attali and G. Pagès, "Approximations of functions by a multilayer perceptron: A new approach," *Neural Netw.*, vol. 10, no. 6, pp. 1069–1081, 1997.
[73] A. K. Pandey and N. K. Goyal, "Predicting fault-prone software module using data mining technique and fuzzy logic," *Int. J. Comput. Commun. Technol.*, vol. 2, nos. 2–4, pp. 56–63, 2010.
[74] S. Chatterjee and B. Maji, "A new fuzzy rule based algorithm for estimating software faults in early phase of development," *Soft Comput.*, vol. 20, no. 10, pp. 4023–4035, Oct. 2015.
[75] K. Li, Z. Cao, L. Cao, and R. Zhao, "A novel semi-supervised fuzzy c-means clustering method," in *Proc. Chin. Control Decis. Conf.*, Guilin, China, 2009, pp. 3761–3765.
[76] B. Gabrys and L. Petrakieva, "Combining labelled and unlabelled data in the design of pattern classification systems," *Int. J. Approx. Reasoning*, vol. 35, no. 3, pp. 251–273, 2014.
[77] A. Bouchachia, "Learning with partly labeled data," *Neural Comput. Appl.*, vol. 16, no. 3, pp. 267–293, 2007.
[78] *Statistics Toolbox Release MATLAB*, MathWorks, Inc., Natick, MA, USA, 2016.
[79] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," in *Proc. Int. Workshop Predictor Models Softw. Eng.*, May 2007, p. 9.
[80] *PROMISE Software Engineering Repository*. Accessed: Mar. 10, 2018. [Online]. Available: http://promise.site.uottawa.ca/SERepository
[81] *Tera-PROMISE Repository*. Accessed: Mar. 10, 2018. [Online]. Available: http://openscience.us/repo/defect/
[82] R. Moser, W. Pedrycz, and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction," in *Proc. IEEE Int. Conf. Softw. Eng.*, May 2008, pp. 181–190.
[83] D. J. Hand, *Construction and Assessment of Classification Rules*. Chichester, U.K.: Wiley, 1997.
[84] Y. Zhou, Y. Yang, B. Xu, H. Leung, and X. Zhou, "Source code size estimation approaches for object-oriented systems from UML class diagrams: A comparative study," *Inf. Softw. Technol.*, vol. 56, no. 2, pp. 220–237, 2014.
[85] K. Pal, R. K. Mudi, and N. R. Pal, "A new scheme for fuzzy rule-based system identification and its application to self-tuning fuzzy controllers," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 4, pp. 470–482, Aug. 2002.
[86] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco, CA, USA: Morgan Kaufmann, 2005.
[87] H. Liu and R. Setiono, "Chi2: Feature selection and discretization of numeric attributes," in *Proc. ICTAI*, 1995, pp. 388–391.

ALI ARSHAD received the B.S. degree in computer science from Iqra University, Pakistan, in 2008, and the M.S. degree in software engineering from International Islamic University, Pakistan, in 2012. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Xidian University, China. His research interests include machine learning, semi-supervised learning, and Fuzzy C-Mean clustering.

SAMAN RIAZ received the M.Sc. and M.Phil. degrees in applied mathematics from Quaid-i-Azam, Pakistan, in 2006 and 2008, respectively. She is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Xidian University, China. Her research interests include machine learning and probability.

LICHENG JIAO (SM'89–F'16) received the B.S. degree from Shanghai Jiao Tong University, Shanghai, China, in 1982, and the M.S. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 1984 and 1990, respectively. Since 1992, he has been a Professor with the School of Electronic Engineering, Xidian University, Xi'an, where he is currently the Director of the Key Laboratory of Intelligent Perception and Image Understanding, Ministry of Education of China. He is also in charge of about 40 important scientific research projects. He has authored or co-authored over 20 monographs and 100 papers in international journals and conferences. His research interests include image processing, natural computation, machine learning, and intelligent information processing.

APARNA MURTHY received the B.E. degree in electronics and communication engineering from Kuvempu University, Shimoga, India, in 1995, and the M.Tech. degree in electronics from Visvesvaraya Technological University, Belgaum, India, 2005. She was a Lecturer with the Department of Electronics and Communication Engineering, BMS Engineering College, from 1998 to 2010. She was involved in the field of programming using C/CCC and MATLAB platform. She is currently with Professional Engineers Ontario, Canada, as an Engineering Intern.