IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 3, 2018, accepted May 8, 2018, date of publication May 10, 2018, date of current version June 5, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2835304

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, Ministry of Education of China, International Joint Collaboration Laboratory of Intelligent Perception and Computation, International Research Center of Intelligent Perception and Computation, Xidian University, Xi'an 710071, China
[4]Professional Engineers Ontario, Toronto, 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 (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 consequential research area in software quality promise. In this paper, we propose a semi-supervised deep fuzzy C-mean (DFCM) clustering for software fault prediction, which is the cumulation of semi-supervised DFCM clustering and feature compression techniques. Deep is utilized for the feature-based multi clusters of unlabeled and labeled data sets along with their labeled classes. In our approach, for the training model, we simultaneously deal with the unsupervised data and supervised data to exploit the obnubilated information from unlabeled data to labeled data to support the construction of the precise model. We utilize DFCM clustering to handle the class imbalance problem and withal fuzzy theory logic is very akin to human logic and it is facile to comprehend. We further ameliorate the prediction performance with the coalescence of feature learning techniques-feature extraction and feature selection (using random-under sampling) to generate good features and remove irrelevant and redundant features to reduce the noisy data for classification. However, by the performance of the model results, the amalgamation of deep multi clusters and feature techniques work better due to their ability to identify and amalgamation essential information in data feature. The classification model is predicted on the maximum homogeneous between the features of labeled and unlabeled data, the model is trained on the un-noisy data set obtained by the deep coalescence of multi clusters and feature techniques. To check the efficacy of our approach, we chose data sets from real-world software project (NASA & Eclipse), and then we compared our approach with a number of latest classical base-line methods, and investigate the performance by using performance measures such as probability of detection, F-measure, and area under the curve.

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

I. INTRODUCTION

Fault prediction is an paramount practice to ameliorate reliability and quality of software entities. It aims at understanding the co-dependencies among variables and processing. The intricacies of software are incrementing day by day for sundry reasons incrementing authoritative ordinances of infusion of incipient technologies, reliability, and security by the users. One possible way to deal with this problem is to prognosticate consequential software quality features during premature phases of software development such as fault-proneness, reliability, testability, endeavor, and maintainability. Identification of software faults afore they authentically make the software fail is kenned as software fault prediction. Many researchers have addressed this problem and sundry software techniques are available for fault prediction [1], [2], [3], [4], [5], [6] and [7].

Supervised learning models are one of the best choices for software fault prediction if only labeled data are provided for training model. Intuitively for the better accuracy of prediction, we required large size of the labeled data for training set [8], [9]. The performance of the prediction could be dramatically reduced with a size of the training set is maximum decrementing. Consequently, for better accuracy one drawback of supervised learning is that the size of training

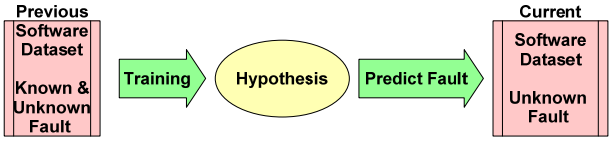

FIGURE 1. Process of Software Fault Prediction.

data set should be large as possible, but it 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

Bennett and Demiriz [10], Joachims [11], & Belkin *et al.* [12] have proposed the semi-supervised approach for classification. However, few researchers have been used simultaneously to exploit the obnubilated information from unlabeled data to labeled data [13], [14], [15] & [16]. Many semi-supervised approaches are used for fault prediction. However, most of them have been dealing with balanced classes [9].

The reason of class imbalance problem occur where some classes are highly underrepresented compared to other classes. In these cases, classifier tends to make more errors on small classes and may even ignore the complexity, although minority class is always more of interest. This problem affects the performance of the model. This problem has gained more attention of researchers, lately. The K-nearest neighbor classifier (KNN) [17], [18] 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 [19], [20]. There are many Fuzzy set theory based algorithms [21], [22], [23], and [24] are proposed for imbalanced classes.

This paper is the extension of our previous work [25], in our previous work; we focused on the preprocessing data technique by feature extraction for the classification of two classes. However, the performance of the classification accuracy is affected with imbalanced classes.

In our approach, the aim to develop the new semi-supervised approach in which both supervised data and unsupervised data are utilized simultaneously during clustering process, in which the obnubilated information is exploited from unlabeled data to support the construction of good classifier. In the field of pattern recognition, the coalesce analysis of labeled data and unlabeled data is very useful.

However, to the best of our knowledge, very few researchers have utilized multi clusters to handle the class imbalance problem [26], 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. In our paper, we utilized Deep Fuzzy C-Mean multi clustering to handle the class imbalance problem and withal fuzzy logic are very akin to human logic and it is facile to comprehend.

The prosperity of prediction model depends strongly on the feature/attributes that are utilized as an input to design the predictor. It is commonly believed, more features do not indispensably avail identification of systems based on input-output data. Utilization of many features conventionally increases the time and cost, and sometime it may result in more hazard by making the system complex. Hence, dimensionality reduction of feature that is usually done by two ways, one is by feature extraction [27], [28] and second is by feature selection [29], [30]. However, few researchers have cumulated feature extraction and feature selection to ameliorate data quality in software fault prediction.

Our main objective is to utilize this coalescence of feature reduction techniques to generate the good features from clusters of all the subsets of supervised and unsupervised data along with labeled classes. Feature selection (Random under-sampling (RUS)) [31] is used to handle the problem of imbalance number of the features in the subset of labeled and unlabeled data set and also remove irrelevant and redundant features to reduce the noisy data for classification. Feature extraction through clustering techniques leads to many issues such as explicated in [32], [33], and [34]. We could handle these issues by feature selection (RUS) technique.

In this paper, we utilized ''Deep'' because of two reasons, one is used for deep correlation between supervised and unsupervised data with multi clusters and second for the deep correlation between Deep Fuzzy C-Mean clustering and feature techniques to find the best input data for the classifier. To best of our knowledge, the development of classifier is based on human-understandable rules depend on the similarity between the features of unlabeled and labeled along with the labeled classes.

Feature technique is utilized to generate a good feature and remover irrelevant and redundant features to reduce noisy data for classification.

The main motivation of DFCM for the classification of software fault prediction on the pre-processing step of semi-supervised multi-clustering to create new features with inhibited labeled data and abundant unlabeled data. A semi-supervised data creates two clusters into unsupervised and supervised that tend to maximize intra-cluster class and intra-cluster features by using FCM clustering.

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

1) To the best of our knowledge, we proposed the new semi-supervised approach, which simultaneously deals with the supervised data and unsupervised data during clustering to exploit the hidden information from unlabeled data.

2) We proposed a novel algorithm using both Deep Fuzzy C-Mean clustering and coalesce feature selection techniques. Deep multi-clustering for the class imbalanced problem and combine feature selection techniques redundancy control for classification.

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

In this section, we briefly introduce the background of software fault prediction, semi-supervised learning, class-imbalance problem, feature learning and sampling technique.

Software fault prediction is very auxiliary to predict the fault of software modules. Many researchers have used a wide variety of machine learning techniques such as decision trees, clustering and SVM [1], [3] & [4].

Semi-supervised learning is a machine learning technique to improve the performance of the model; the model is trained by utilizing few labeled with abundant unlabeled data. Semi-supervised approach for software fault prediction is studied by Nigam *et al.* [35], the algorithm utilized in speech processing computational linguistics. Here, a set of untagged data is used and steps of collocation labeling. Using this labeled data is trained for partitioning, iteratively on the probability of co-occurrence till the data grows and reduces the untagged set. Once the grouping is complete, then classifier is used. The algorithm is dependent on the collocational list of entries [35].

Liu *et al.* [7] proposed a two-stage data preprocessing approach for software fault prediction. 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 random under-sampling technique to keep the balance between the faulty and non-faulty classes.

Riloff *et al.* [36] proposed the modification of self-training algorithm for the semi-supervised approach for software fault prediction. It is a two-step process one is a utilization of Multidimensional Scaling (MDS) and other is semi-supervised learning algorithm for fitting the confidence intervals to fit the estimated values.

Li *et al.* [37] 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.

Lu *et al.* [38] 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

In feature learning Coates *et al.* [39], 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.

Lu *et al.* [40] proposed the semi supervised learning approach for the defect prediction, embedded a preprocessing strategy, Multi-dimension scaling (MDS), they shows that the integration of the Fitting–the-confident-Fits (FTcF) with MDS performs better than supervised learning algorithm.

Catal [41] analysis the performance of fault prediction by compared four semi-supervised classification method for including Class Mass Normalization (CMN) methods, Low-Density Separation (LDS), Support Vector Machine (SVM) and Expectation-Maximization (EM-SEM). According to this comparison, when the data size is large, LDS algorithm performs better compared to SVM.

Zhang *et al.* [42] used Non-negative Sparse Graph-based Labeled Propagation approach (NSGLP) for software defect classification and prediction. According to this result, NSGLP performs better than LDS when the dataset is unbalanced.

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 [43] 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 Maji [44] also use fuzzy if the fault in software requirement analysis phase.

It is known that good modeling tool is the cognation 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) [27], [28] and features selection [29], [30]. The data preprocessing plays important role to improve the quality of software datasets [45], [46], [47] which include feature selection and reduction (or sampling). Khoshgoftaar *et al.* [51] combined filter based feature ranking methods and random under-sampling for improved the data preprocessing.

Gabry's and Petrakieva [48] or Bouchachia [49], they proposed the method to ameliorate the classification accuracy with very few labeled and abundant unlabeled samples are available, they used semi-supervised approach in which during the clustering process they deal with labeled and unlabeled data simultaneously.

Cai *et al.*[15] 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.

Here, we used Fuzzy C-Mean with deep multi clusters for feature extraction utilizing both labeled and

unlabeled data, simultaneously, during clustering process. For the imbalanced number of features in the labeled set and unlabeled set, we used random under-sampling [51].

III. SEMI-SUPERVISED DEEP FUZZY-C MEANS CLUSTERING

In this section, we present a human-interpretable learning-based model for software fault prediction, which cumulates the DFCM clustering with multi-feature techniques and classification based on similarity among the selected features of labeled data with their classes and unlabeled data. The aim of the proposed method is to get a best classification model with few labeled data with imbalanced classes, which would improve the performance of fault prediction.

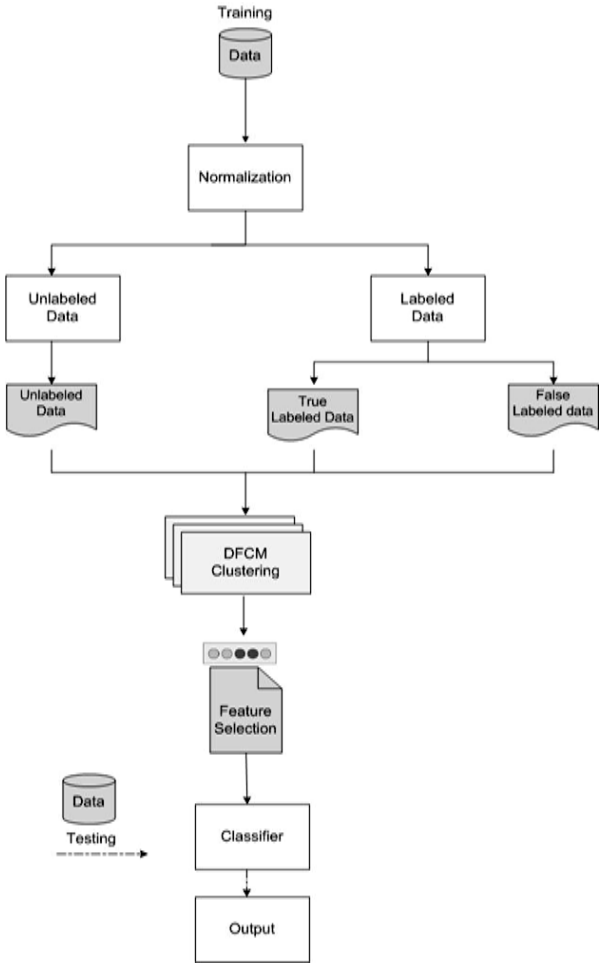

FIGURE 2. Flowchart of Deep Fuzzy C-Mean Clustering (DFCM).

A. FRAMEWORK OF OUR APPROACH

Fig 2 gives the framework of our DFCM clustering approach. In this framework, after normalization of semi-supervised data using min-max approach, the data converted into two subsets of labeled and unlabeled datasets (supervised and unsupervised) in the first layer. In the second layer, the supervised set split into two subsets based on the classes (true and false) along with their features. In the next step, two supervised subsets and one unsupervised set split into

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. Construct membership matrices $U_{TL}, U_{FL} \& U_{UN}$ with random decimal fraction.
2. Compute cluster center $V_{TL}^{(k)}, V_{FL}^{(k)} \& V_{UN}^{(k)}$ using formula of cluster center of FCM [32].
3. Update $U_{TL}, U_{FL} \& U_{UN}$ using formula of membership of FCM [37].
4. Repeat step 2 & 3 until $\left\| \mathrm{J}^{(\mathrm{t})} - \mathrm{J}^{(\mathrm{t}-1)} \right\| < \varepsilon$ for all labeled and unlabeled subsets separately.

k-clusters based on features, we have total k (2+1) clusters in the third layer, calculate DFCM membership and DFCM centroid by algorithm 1.

In next step feature extraction of k (2+1) cluster by DFCM. Before the classification, we select s features by RUC [51] based on ranking to balance the number of features between supervised and unsupervised subsets, where s is the minimum number of features in any subsets. The detail of the feature extraction and classification are going to be discussed in next section. Examples of clusters and classification are given in figure 3.

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 reduce the accumulated the design for decision-making system. There are two main broad ways to reduce the feature space i.e. feature selection [24], [30] and feature extraction [27], [28].

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.

For the feature extraction, 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.

$$f_k(x) = \max(0, \mu(z) - z_k) \tag{1}$$

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

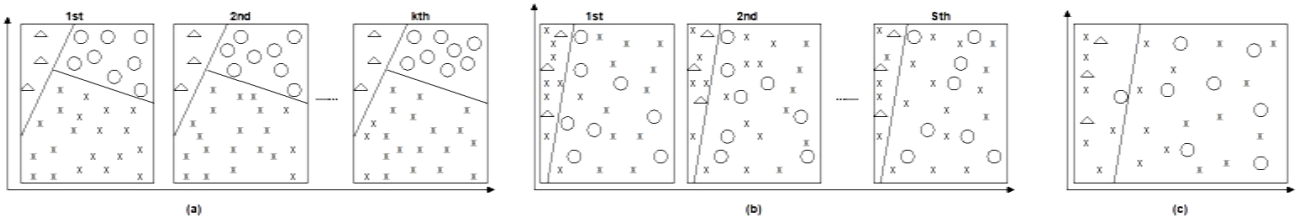

FIGURE 3. Example of clustering and classification Δ Labeled data class1/true, ○ Labeled data class2/false, x unlabeled data. (a) Shows, k is the number of features and every feature make one cluster. Every feature has three more clusters (labeled data class1/true, labeled data class2/false, and unlabeled data). (b) Shows, "s" is the number of selected features/clusters by (RUS) from all three subsets (labeled data class1/true, labeled data class2/false, and unlabeled data), In pre-classification the unlabeled data belong to labeled class1 or labeled class2 according to the maximum similarity between the features. (c) Shows, final classification is the re-union of the clusters which is based on the maximum average of the maximum similarity between all selected features (clusters).

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

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.* [51], where "s" are the number of minimum features in any subset. To measure the similarity between the pair of features of labeled classes and unlabeled data set. We choose the Euclidean distance between "s" cluster centers of labeled classes and unlabeled data points with s features.

$$max\ Sim_j\left(x, V_L^{(s)}\right) = min\left|x_i - V_{jL}^{(s)}\right|, j \in \text{True\&False} \tag{2}$$

Where $x_i \in X_{UN}$, V_{jL} is the set of the centroid of labeled class1 (True) and label class2 (False) and "s" is the number selected features in all subsets. With every selected feature clusters, find the maximum similarity by using equation 2 between the features of unlabeled data and labeled data (class1 or class2). In the final classification step, followed by algorithm 3. Find the maximum average of the maximum similarity between the selected features of unlabeled data and labeled data. Then, unlabeled data point corresponding to maximum average adding in the particular labeled class (True or False).

Algorithm 3 Classifier of DFCM

Input:

The data set $X = \{x_{F+1}, x_{F+2} \ldots, x_n, l\}$, with s selected features, $V_{TL}^{(s)}$, $V_{FL}^{(s)} \& V_{UN}^{(s)}$, $l \in [T, F]$.

Output:

Predicted labeled data $Y = \{y_{l+1}, y_{l+2} \ldots, y_n$

1. Y = ∅
2. **For** $i = l + 1$ **to** $n \& j = T, F$ **do**
   Computing $maxSim_j\left(x, V_L^{(s)}\right)$ by using equation 1,
3. **If** $\max\_avg\_max\ Sim_j\left(x, V_L^{(s)}\right) \in$ class True
4. Adding x_i into class True otherwise
5. Adding x_i into class False.
6. Updating all rest data points in X into Y
7. **return** Y.

IV. EXPERIMENT

A. DATA PREPARATION

In this paper, MATLAB 2016a [50] is used as the programming tool. In order to verify the clustering performance of the DFCM algorithm on ten NASA datasets (cm1, jm1, kc1, kc3, mc2, mw1, pc1, pc3, pc4, and pc5) [52], [53] and three Eclipse dataset (Eclipse 2.0, Eclipse 2.0 and Eclipse 3.0) [54], [55], are used to test the experiment. All datasets with 10%, 20%, and 30% rate of labeled data and contain 2 classes true and false. We select objective threshold 0.1 which is used to stop the iteration for updating new cluster centers for all datasets, and degree of fuzziness m = 2.

Table 1 shows the benchmark NASA and Eclipse dataset 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.

TABLE 1. NASA and eclipse dataset.

Dataset	No. Of Samples	No. Of Features	No. Of Faulty Modules	No. of Non-Faulty Modules	No. of Classes
cm1	327	37	42	285	2
jm1	7782	21	1672	6110	2
kc1	2109	20	326	1783	2
kc3	194	39	36	158	2
mc2	125	39	44	81	2
mw1	253	37	27	226	2
pc1	705	37	61	644	2
pc3	1077	37	134	943	2
pc4	1458	37	178	1280	2
pc5	17186	38	516	16670	2
Eclipse 2.0	6729	155	975	5714	2
Eclipse 2.1	7888	155	854	7034	2
Eclipse 3.0	10593	155	1568	9025	2

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))

B. PERFORMANCE MEASURE

In table 2, Confusion matrix is used to evaluate the performance by ROC-curve of the model by using area under the curve (AUC). 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.

We use three evaluation measures to check the performance on different compared methods, namely Probability of detection (Pd), F-measure, and AUC. They are defined as follows.

1 $\text{Pd} = \text{n(T P)}/(\text{n(T P)} + \text{n(FN)})$.

2 $\text{Precision} = \text{n(T P)}/(\text{n(T P)} + \text{n(FP)})$.

3 $\text{F} - \text{measure} = 2(\text{Pd})(\text{precision})/ (\text{pd} + \text{precision})$

In machine learning, AUC are widely used to check the performance evaluation spatially for imbalanced classes.

C. EXPERIMENTAL DESIGN

In this section, we design experiments to demonstrate the efficacy of our proposed approach (DFCM) for software fault prediction. This study seeks to understand.

1. The benefit of utilizing supervised data and unsupervised data simultaneously during the multi-clustering to create the new features to train the classification model.
2. Impact of deep multi clustering on imbalanced classes.
3. Impact of feature extraction and feature selection for classification.

Semi-supervised data prediction into supervised (labeled) and unsupervised (unlabeled) in the first layer. In second layer supervised data split into two subsets for each class (True and False). k clusters are created for each class of supervised data and unsupervised data. In the third layer, calculate the centroid and membership for each cluster of all subsets of supervised and unsupervised by using DFCM. In the fourth hidden layer using algorithm 2 as activation function for feature extraction. 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. For balance, the number of features in all subsets of supervised and unsupervised date, select ''s'' features, where ''s'' is the minimum number of features extracted in any subset. On the classification stage based the unlabeled data is predicted as labeled on basis of similarity between the features of labeled classes (true and false) and unlabeled data sets.

TABLE 3. Details of selected features.

Dataset	No of feature extraction in all subsets (True, False, Unlabeled)	s (No of selected features in all subsets)	Percentage of selected features
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
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
Avg			0.515

To investigate the performances of our approach on thirteen data sets by using three performance measures (Pd, F-measure, and AUC), each result is the average of 100 runs. We compare our approach with five methods, which are proposed in the last five years that are FTF [38], LDS [41], [57], CMN [41], [58], NTC (NB) [7], and NSGLP [42]. Brief analysis of DFCM and other compared methods are shown in next section.

D. EXPERIMENTAL RESULTS AND ANALYSIS

We show the performances of our method with other methods on the basis of thirteen datasets, all results are the average of 100 runs.

Table 4, 5, and 6, summarizes the result of Pd of DFCM with other approaches on thirteen datasets with 10%, 20% and 30% labeled rates, we examine that the DFCM approach has the best improvement over other algorithms on all datasets although the labeled rate is low.

In figure 4, the comparison of the Pd of DFCM approach with other approaches on the average of thirteen datasets. From the multiple bar charts, we can observe that the results of NSGLP and DFCM are higher on the average of ten datasets with all labeled rates. We can analyze that the performance of algorithms can be improved by utilizing supervised data and unsupervised data simultaneously for training model.

TABLE 4. Pd of DFCM with compared methods using NASA And eclipse dataset at labeled rate = 0.1.

Datasets	Algorithms					
	FTF	LDS	CMN	NTC (NB)	NSGLP	DFCM
cm1	0.37	0.50	0.46	0.49	0.71	**0.74**
jm1	0.44	0.54	0.46	0.55	0.69	**0.73**
kc1	0.42	0.60	0.45	0.47	0.68	**0.69**
kc3	0.40	0.47	0.44	0.50	0.66	**0.69**
mc2	0.52	0.52	0.54	0.60	0.74	**0.77**
mw1	0.25	0.35	0.28	0.50	0.60	0.59
pc1	0.28	0.31	0.33	0.43	0.62	**0.63**
pc3	0.33	0.36	0.36	0.43	0.67	**0.67**
pc4	0.46	0.47	0.51	0.55	0.70	**0.72**
pc5	0.53	0.32	0.54	0.72	0.75	**0.78**
Eclipse 2.0	0.42	0.65	0.46	0.46	0.62	**0.77**
Eclipse 2.1	0.50	0.42	0.44	0.50	0.60	**0.72**
Eclipse 3.0	0.39	0.40	0.47	0.59	0.69	0.67
Avg	0.41	0.45	0.44	0.52	0.67	**0.71**

TABLE 5. Pd of DFCM with compared methods using NASA and eclipse dataset at labeled rate = 0.2.

Datasets	Algorithms					
	FTF	LDS	CMN	NTC (NB)	NSGLP	DFCM
cm1	0.39	0.50	0.49	0.50	0.72	**0.76**
jm1	0.45	0.55	0.51	0.56	0.70	**0.74**
kc1	0.46	0.61	0.48	0.51	0.71	**0.73**
kc3	0.43	0.49	0.49	0.50	0.69	**0.70**
mc2	0.55	0.58	0.56	0.61	0.79	**0.79**
mw1	0.27	0.36	0.30	0.53	0.63	**0.65**
pc1	0.31	0.45	0.35	0.45	0.66	0.65
pc3	0.35	0.46	0.39	0.44	0.72	**0.73**
pc4	0.49	0.50	0.54	0.55	0.74	**0.75**
pc5	0.55	0.33	0.61	0.73	0.78	**0.78**
Eclipse 2.0	0.51	0.53	0.41	0.43	0.66	**0.73**
Eclipse 2.1	0.46	0.49	0.50	0.53	0.65	**0.68**
Eclipse 3.0	0.40	0.45	0.56	0.64	**0.79**	**0.79**
Avg	0.43	0.48	0.48	0.54	0.71	**0.73**

TABLE 6. Pd of DFCM with compared methods using NASA and eclipse dataset at labeled rate = 0.3.

Datasets	Algorithms					
	FTF	LDS	CMN	NTC (NB)	NSGLP	DFCM
cm1	0.42	0.51	0.54	0.51	0.76	**0.78**
jm1	0.52	0.56	0.55	0.58	0.73	**0.77**
kc1	0.49	0.62	0.52	0.55	0.75	**0.79**
kc3	0.47	0.51	0.48	0.51	0.70	**0.72**
mc2	0.58	0.60	0.59	0.61	0.83	**0.84**
mw1	0.30	0.36	0.31	0.55	0.67	**0.71**
pc1	0.35	0.49	0.38	0.47	0.69	**0.69**
pc3	0.38	0.41	0.43	0.44	0.74	**0.74**
pc4	0.54	0.56	0.58	0.56	0.78	**0.81**
pc5	0.57	0.35	0.63	0.74	0.82	**0.83**
Eclipse 2.0	0.49	0.48	0.57	0.60	0.79	**0.84**
Eclipse 2.1	0.47	0.46	0.46	0.56	0.78	**0.79**
Eclipse 3.0	0.52	0.58	0.51	0.55	0.67	**0.82**
Avg	0.47	0.50	0.50	0.56	0.75	**0.78**

Table 7, 8, and 9, shows the F-measure values of DFCM and other approaches on thirteen datasets with all labeled rates. The result shows that DFCM virtually outperforms the other approaches for software fault prediction. The average of F-measure of DFCM is highest with all labeled rates.

In figure 5, we can observe from the comparison of F-Measure by multiple bar-charts that the performance of

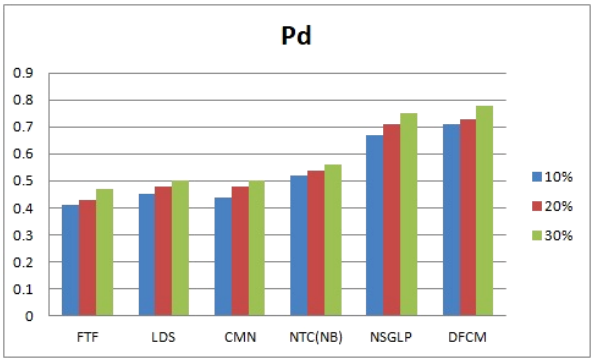

FIGURE 4. Comparison of Pd of DFCM with other algorithms.

TABLE 7. F-measure of DFCM with compared methods using NASA and eclipse dataset at labeled rate = 0.1.

Datasets	Algorithms					
	FTF	LDS	CMN	NTC (NB)	NSGLP	DFCM
cm1	0.30	0.34	0.32	0.34	0.37	**0.38**
jm1	0.38	0.39	0.39	0.42	0.43	**0.44**
kc1	0.37	0.42	0.39	0.41	0.44	0.43
kc3	0.33	0.35	0.35	0.40	0.40	**0.41**
mc2	0.44	0.49	0.46	0.51	0.57	**0.58**
mw1	0.22	0.27	0.23	0.31	0.34	**0.36**
pc1	0.23	0.28	0.26	0.31	0.35	0.34
pc3	0.25	0.29	0.26	0.30	0.34	**0.36**
pc4	0.36	0.41	0.37	0.44	0.49	**0.51**
pc5	0.45	0.50	0.46	0.53	0.59	**0.61**
Eclipse 2.0	0.40	0.44	0.39	0.40	0.44	**0.46**
Eclipse 2.1	**0.42**	0.38	0.35	**0.42**	0.37	**0.42**
Eclipse 3.0	0.33	0.36	0.31	**0.41**	0.40	**0.41**
Avg	0.34	0.38	0.35	0.40	0.43	**0.44**

TABLE 8. F-measure of DFCM with compared methods using NASA and eclipse dataset at labeled rate = 0.2.

Datasets	Algorithms					
	FTF	LDS	CMN	NTC (NB)	NSGLP	DFCM
cm1	0.32	0.35	0.32	0.33	0.37	**0.39**
jm1	0.39	0.45	0.42	0.43	0.44	**0.45**
kc1	0.39	0.42	0.41	0.41	0.45	**0.44**
kc3	0.36	0.39	0.38	0.41	0.41	**0.43**
mc2	0.47	0.51	0.49	0.53	0.60	**0.63**
mw1	0.21	0.28	0.24	0.32	0.37	**0.40**
pc1	0.25	0.32	0.27	0.31	0.39	**0.40**
pc3	0.28	0.33	0.30	0.35	0.39	**0.41**
pc4	0.39	0.45	0.40	0.46	0.53	**0.55**
pc5	0.45	0.48	0.47	0.54	0.58	**0.60**
Eclipse 2.0	0.41	0.42	0.35	0.35	0.39	**0.44**
Eclipse 2.1	0.40	0.40	0.39	**0.46**	0.37	0.40
Eclipse 3.0	0.38	0.37	0.42	0.54	0.53	**0.56**
Avg	0.36	0.40	0.37	0.42	0.45	**0.47**

DFCM is better than other studied models. From the lowest results of CMN approach, we can analyze that the class imbalanced problem cannot be ignored for the higher accuracy of classification.

Table 10, 11, and 12 are the results of AUC to check the performance of the model on the class-imbalanced dataset. According to the results, the average of AUC is highest than other approaches.

TABLE 9. F-measure of DFCM with compared methods using NASA and eclipse dataset at labeled rate = 0.3.

Datasets	Algorithms					
	FTF	LDS	CMN	NTC (NB)	NSGLP	DFCM
cm1	0.32	0.35	0.34	0.35	0.38	**0.40**
jm1	0.40	0.45	0.42	0.45	0.46	**0.48**
kc1	0.41	0.45	0.43	0.43	0.47	**0.48**
kc3	0.37	0.41	0.37	0.41	0.43	**0.45**
mc2	0.51	0.55	0.52	0.55	0.62	**0.65**
mw1	0.23	0.26	0.22	0.32	0.36	**0.40**
pc1	0.27	0.35	0.30	0.34	0.41	**0.42**
pc3	0.31	0.37	0.33	0.40	0.42	**0.43**
pc4	0.42	0.48	0.43	0.48	0.55	**0.56**
pc5	0.46	0.52	0.46	0.55	0.59	**0.62**
Eclipse 2.0	0.41	0.35	0.43	0.48	0.59	**0.60**
Eclipse 2.1	0.37	0.41	0.35	0.45	**0.55**	0.54
Eclipse 3.0	0.39	0.51	0.44	0.41	0.47	**0.59**
Avg	0.37	0.42	0.39	0.43	0.48	**0.51**

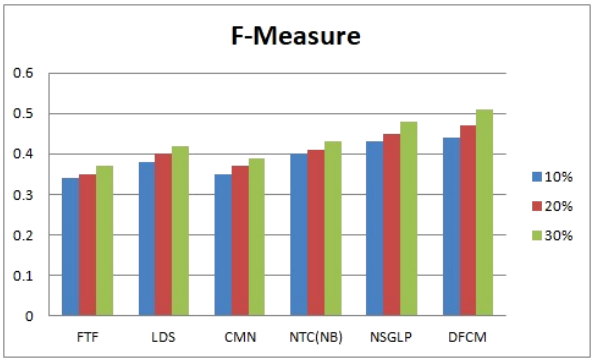

FIGURE 5. Comparison of F-Measure of DFCM with other algorithms.

TABLE 10. AUC of DFCM with compared methods using NASA and eclipse dataset at labeled rate = 0.1.

Datasets	Algorithms					
	FTF	LDS	CMN	NTC (NB)	NSGLP	DFCM
cm1	0.62	0.71	0.67	0.71	0.75	**0.84**
jm1	0.65	0.71	0.66	0.73	0.72	**0.81**
kc1	0.72	0.78	0.76	0.78	0.82	**0.85**
kc3	0.74	0.77	0.72	0.79	0.84	**0.87**
mc2	0.85	0.93	0.88	0.70	0.95	0.93
mw1	0.66	0.64	0.65	0.71	0.73	**0.76**
pc1	0.69	0.75	0.61	0.72	0.79	**0.82**
pc3	0.72	0.74	0.73	0.74	0.76	**0.80**
pc4	0.80	0.83	0.84	0.81	0.86	**0.89**
pc5	0.88	0.91	0.89	0.93	0.95	**0.97**
Eclipse 2.0	0.79	0.81	0.71	0.79	0.81	**0.82**
Eclipse 2.1	**0.78**	0.75	0.73	0.74	0.76	0.77
Eclipse 3.0	0.74	0.76	0.76	0.77	0.78	**0.80**
Avg	0.74	0.78	0.74	0.76	0.81	**0.84**

TABLE 11. AUC of DFCM with compared methods using NASA and eclipse dataset at labeled rate = 0.2.

Datasets	Algorithms					
	FTF	LDS	CMN	NTC (NB)	NSGLP	DFCM
cm1	0.65	0.73	0.68	0.75	0.79	**0.84**
jm1	0.67	0.76	0.69	0.75	0.74	**0.81**
kc1	0.73	0.79	0.76	0.80	0.85	**0.86**
kc3	0.75	0.83	0.76	0.80	0.86	**0.87**
mc2	0.87	0.93	0.90	0.73	0.95	**0.95**
mw1	0.67	0.71	0.68	0.74	0.76	**0.80**
pc1	0.73	0.80	0.75	0.73	0.84	**0.85**
pc3	0.80	0.78	0.75	0.76	0.83	**0.83**
pc4	0.85	0.86	0.84	0.83	0.90	**0.92**
pc5	0.90	0.84	0.92	0.94	0.96	**0.97**
Eclipse 2.0	0.80	0.79	0.69	0.73	0.81	**0.82**
Eclipse 2.1	0.79	**0.83**	0.76	0.76	0.80	0.81
Eclipse 3.0	0.75	0.84	0.84	0.83	0.83	**0.89**
Avg	0.77	0.81	0.77	0.78	0.84	**0.86**

TABLE 12. AUC of DFCM with compared methods using NASA and eclipse dataset at labeled rate = 0.3.

Datasets	Algorithms					
	FTF	LDS	CMN	NTC (NB)	NSGLP	DFCM
cm1	0.67	0.73	0.69	0.77	0.82	**0.86**
jm1	0.68	0.78	0.70	0.78	0.81	**0.84**
kc1	0.75	0.82	0.78	0.84	0.88	**0.90**
kc3	0.77	0.84	0.79	0.84	0.89	**0.91**
mc2	0.90	0.95	0.91	0.73	0.96	**0.97**
mw1	0.69	0.72	0.69	0.77	0.78	**0.82**
pc1	0.76	0.84	0.79	0.80	0.87	**0.89**
pc3	0.81	0.82	0.78	0.82	0.86	**0.89**
pc4	0.88	0.87	0.86	0.87	0.94	**0.96**
pc5	0.90	0.95	0.92	0.95	0.97	**0.98**
Eclipse 2.0	0.80	0.90	0.85	0.85	0.92	**0.95**
Eclipse 2.1	0.79	0.85	0.81	0.80	0.89	**0.91**
Eclipse 3.0	0.75	0.80	0.76	0.83	0.83	**0.88**
Avg	0.78	0.84	0.79	0.82	0.88	**0.90**

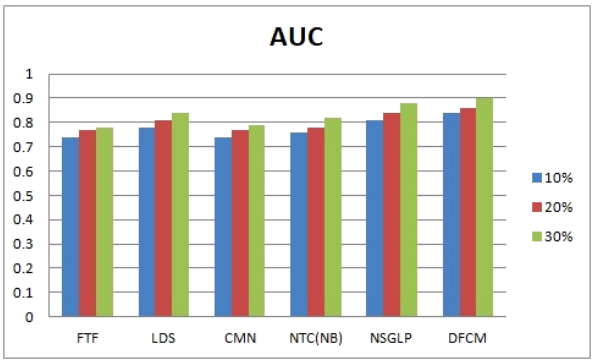

FIGURE 6. Comparison of AUC of DFCM with other algorithms.

Fig 6 is the comparison of DFCM with other algorithms. We can analyze that DFCM outperforms other proposed approaches with all labeled rates. The results are almost same as F-Measure, Hence the class imbalanced problem affects the AUC results.

We conclude three facts from the above results.

1) By comparing DFCM, NTC (NB) and NSGLP with CMN (ignoring class imbalance problem), we observe that due to ignoring the class imbalance problem, the performance of CMN is worse than DFCM, NTC (NB) and NSGLP.
2) By comparing FTF with other approaches, the performance of FTF is worse because FTF is using supervised data for training model. By this comparison, we can conclude that semi-supervised data for training model increases the value of performance measure.

3) By using feature reduction (feature extraction and feature selection) increases the accuracy of classification.

E. STATISTICAL ANALYSIS

We carried out one way ANOVA test to determine the statistical significance to observed performance results. The test was applied at $\alpha = 0.05$ significance level.
The testing hypotheses are

$\mathbf{H_o}$: The performance is same among the six algorithms across the datasets.

$\mathbf{H_a}$: At least one of the algorithms performances is significantly better than the other algorithms.

TABLE 13. ANOVA test for Pd at 10% labeled data.

	df	Sum square	Mean square	F-Value	P-Value
Algorithms	5	1.0275	0.2055	33.4757	1.1102 e-16
Residuals	72	0.4420	0.0011		

In table 13, an example of one way ANOVA test for the Pd with 10% labeled data. The classification results between ten different datasets of experiments that use only one algorithm. An immense F-value denotes that the outcome of different approaches varies more than the outcomes of any concrete algorithm. The p-value is a probability of observing a test statistic as extreme as the one authentically observed. The more minute the p-value, more vigorously the tests reject the null hypothesis.

In our example, the p-value 1.1102 e-16 is much smaller than α, the result is null hypothesis $\boldsymbol{H_o}$ is rejected. We can conclude that there is at least one approach amongst six approaches as significantly outperforms the others.

TABLE 14. P-Value of ANOVA test for all performance measures.

Size of Label data	F- Measure	Pd	AUC
0.1	0.0069	1.1102 e-16	0.0043
0.2	0.0048	1.1103 e-15	0.0012
0.3	0.0002	1.1100 e-16	2.3426 e-05

In table 14, the overall results of all performance measure (Pd, AUC, and F-measure) of ANOVA test on all size of labeled data are presented. Those outcomes are significant where the p-value is less than the significance level, these values are presented in bold font. We observed that all p-values of three performance measures are less than the significance level (across all thresholds) at least one of the proposed approaches significantly outperforms the others of correct classification in fault detection.

Next for analysis the significance comparison between all proposed approaches, we performed the post-hoc test using Turkey Honestly Significantly Difference (HSD) method [49]. For AUC and F-measure, except with 30% of labeled data we did not obtain a significant difference between all discussed approaches. Hence, we will only focus on the probability of detection (Pd) with 10%, 20% and 30% labeled rates. Table 15 shows the results of Turkey's HSD pairwise comparison among discussed approaches. If the intersection between the two modeling approaches indicates "All" this designates that there are significant prediction performance differences for all size of labeled rate. The result "None" has inverse meaning. We can analyze which algorithm is performing better from the result reported in table 15, but the ascendancy of DFCM is quite apparent.

TABLE 15. Significance comparison of Pd.

	FTF	LDS	CMN	NTC (NB)	NSGLP
LDS	None	-	-	-	-
CMN	None	None	-	-	-
NTC(NB)	10%, 20%	None	10%	-	-
NSGLP	All	All	10%, 30%	10%, 20%	-
DFCM	All	All	All	All	10%, 30%

From table 15, we can observe that DFCM significantly outperforms FTF, LDS, CMN, NTC (NB), and NSGLP for all size of the labeled rate except NSGLP for 20% labeled rate. We achieve better performance for different size of the labeled rate because our approach incorporates simultaneously labeled data and unlabeled data in the learning process.

V. THREATS TO VALIDITY

Our experimental results might be affected by some threats to validity.

A. CONSTRUCT VALIDITY

These threats refer to the approximateness of our evaluation measure. A first threat to the validity of our work is that we assume that all the faults that we utilized in the experiment had same weights. We make utilization of Probability of detection (Pd), F-measure and AUC to evaluate the software fault prediction (SFP). AUC has lower variance, and is more reliable to indicate the predictive potential of the methods when compared with other performance measure, such as precision, recall or F-Measure. Finally Post-hoc test using Turkey Honestly, significantly difference (HSD) method [49] to further validate the significance of the differences in performance.

B. INTERNAL VALIDITY

These threats refer to experimater biases. To avoid this type of threat, all the implementation is cross-checked by our researcher group. Withal, we perform our experiment 100 times and report the average performance over 100 runs. Moreover, the datasets are carefully examined, whether non-numeric features are eliminated. Thus, we believe there are minimal threats to internal validity.

C. EXTERNAL VALIDITY

These threats refer to the generalizability of our experimental results. To ensure the representativeness of our experiment, we used NASA and Eclipse dataset which are commonly used for software fault prediction. In addition, we choose Fuzzy C-Mean clustering as a base method and random under-sampling for feature selection, which are widely used in software fault prediction, to ensure the soundness of the results.

VI. CONCLUSION

In this paper, we provided a deep approach, DFCM which incorporate the semi-supervised Deep Fuzzy C-Mean clustering and feature compression technique to amend the quality of software dataset utilized by classification model for software fault prediction with imbalanced classes. In this paper, we presented how ''Deep'' multiple clusters can be amalgamated with feature reduction techniques and how it can avail to incorporate simultaneously unlabeled data and labeled data into the training process and withal handle the class imbalance problem.

Coalescence of feature reduction techniques, we can design good prediction system by generate good features and remove irrelevant and redundant features to reduce the noisy data for training classification. In our experiment, we compared our approach with several state-of-the-art semi-supervised software fault prediction approaches. Experiment results demonstrate the potential of our approach in enhancing prediction performance on ten NASA and three Eclipse data sets. The proposed method has the best Pd values and withal the average of both AUC and F-Measure values significantly improved. The post-hoc test using Turkey's (HSD), experiment result shows that the difference between DFCM and compared approaches are statistically significant.

In our future work, we plan to extend our approach in several ways. First, investigate the inter-relations between the feature techniques. Second, study how the characteristic of the multiple clustering results utilized to find the new features in the method influence the classification accuracy. Determinately, empirical studies of our approach with other classification methods to authenticate the generalization of the approach.

REFERENCES

[1] 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.
[2] N. Nagappan and T. Ball, ''Static analysis tools as early indicators of pre-release defect density,'' in *Proc. Int. Conf. Softw. Eng.*, St. Louis, MO, USA, 2005, pp. 580–586.
[3] 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.
[4] 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.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] K. P. Bennett and A. Demiriz, ''Semi-supervised support vector machines,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 11. 1999, pp. 368–374.
[11] T. Joachims, ''Transductive inference for text classification using support vector machines,'' in *Proc. Int. Conf. Mach. Learn.*, 1999, pp. 200–209.
[12] 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, Jan. 2006.
[13] X. Ao *et al.*, ''Combining supervised and unsupervised models via unconstrained probabilistic embedding,'' *Inf. Sci.*, 257, pp. 101–114, Feb. 2014.
[14] S. Basu, A. Banerjee, and R. J. Mooney, ''Semi-supervised clustering by seeding,'' in *Proc. Int. Conf. Mach. Learn.*, 2002, pp. 27–34.
[15] W. Cai, S. Chen, and D. Zhang, ''A simultaneous learning framework for clustering and classification,'' *Pattern Recognit.*, vol. 42, no. 7, pp. 1248–1286, 2009.
[16] 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, no. 1, pp. 331–366, 2005.
[17] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York, NY, USA: Wiley, 1973.
[18] 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.
[19] V. Wu, ''Top 10 algorithms in data mining,'' *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008.
[20] R. Jensen and C. Cornelis, ''Fuzzy-rough nearest neighbor classification,'' in *Transactions on Rough Sets XIII*, J. Peters, A. Skowron, C. Chan, J. Grzymala-Busse, and W. Ziarko, Eds. Berlin, Germany: Springer, 2011, pp. 56–72.
[21] R. Bhatt and M. Gopal, ''FRCT: Fuzzy-rough classification trees,'' *Pattern Anal. Appl.*, vol. 11, no. 1, pp. 73–88, 2008.
[22] R. Jensen and C. Cornelis, ''Fuzzy-rough nearest neighbour classification and prediction,'' *Theor. Comput. Sci.*, vol. 412, no. 42, pp. 5871–5884, 2011.
[23] 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.
[24] 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.
[25] 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. Mechatron. Eng. Conf. (ITOEC)*, Chongqing, China, Oct. 2017, pp. 365–370.
[26] 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.
[27] 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.
[28] 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.
[29] 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.
[30] 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.
[31] C. Cardie, ''Using decision trees to improve case-based learning,'' in *Proc. 10th Int. Conf. Mach. Learn.*, Amherst, MA, USA, 1993, pp. 25–32.

[32] 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.
[33] N. R. Pal and S. Saha, "Simultaneous structure identification and fuzzy rule generation for Takagi–Sugeno models," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 38, no. 6, pp. 1626–1638, Dec. 2008.
[34] R. Nikhil, K. Pal, J. C. Bezdek, and T. A. Runkler, "Some issues in system identification using clustering," in *Proc. Int. Conf. Neural Netw.*, vol. 4. Jun. 1997, pp. 2524–2529.
[35] 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.
[36] E. Riloff, J. Wiebe, and T. Wilson, "Learning subjective nouns using extraction pattern bootstrapping," in *Proc. Conf. Natural Lang. Learn.*, 2003, pp. 25–32.
[37] K. Li, Z. Cao, L. Cao, and R. Zhao, "A novel semi-supervised fuzzy c-means clustering method," in *Proc. Chin. Control Decision Conf.*, Guilin, China, 2009, pp. 3761–3765.
[38] 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.
[39] A. Coates, H. Lee, and Y. Andrew Ng, "An analysis of single-layer network in unsupervised feature learning," in *Proc. 14th Int. Conf. Artif. Intell. Stat. (AISTATS)*, vol. 15. 2011, pp. 215–223.
[40] H. Lu, B. Cukic, and M. Culp, "Software defect prediction using semi-supervised learning with dimension reduction," in *Proc. 27th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, Sep. 2012, pp. 314–317.
[41] C. Catal, "A comparison of semi-supervised classification approaches for software defect prediction," *J. Intell. Syst.*, vol. 23, no. 1, pp. 75–82, 2014.
[42] Z. W. Zhang, X. Y. Jing, and T. J. Wang, "Label propagation based semi-supervised learning for software defect prediction," *Autom. Softw. Eng.*, vol. 24, no. 1, pp. 47–69, 2017.
[43] 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.
[44] S. Chatterjee and B. Maji, "A new fuzzy rule based algorithm for estimating software faults in early phase of development," *Soft Comput.*, vol. 19, pp. 1–13, Jun. 2015.
[45] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, "Choosing software metrics for defect prediction: An investigation on feature selection techniques," *Softw.-Pract. Exper.*, vol. 41, no. 5, pp. 579–606, 2011.
[46] 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.
[47] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[48] B. Gabrys and L. Petrakieva, "Combining labelled and unlabelled data in the design of pattern classification systems," *Int. J. Approx. Reason.*, vol. 35, no. 3, pp. 251–273, 2014.
[49] A. Bouchachia, "Learning with partly labeled data," *Neural Comput. Appl.*, vol. 16, no. 3, pp. 267–293, 2007.
[50] *Statistics Toolbox Release 2016a*, MATLAB, The MathWorks, Inc., Natick, MA, USA, Mar. 2016.
[51] T. M. Khoshgoftaar, C. Seiffert, J. V. Hulse, A. Napolitano, and A. Folleco, "Learning with limited minority class data," in *Proc. Int. Conf. Mach. Learn. Appl.*, 2007, pp. 348–353.
[52] *PROMISE Software Engineering Repository*. Accessed: Dec. 2017. [Online]. Available: http://promise.site.uottawa.ca/SERepository
[53] *Tera-PROMISE Repository*. Accessed: Dec. 2017. [Online]. Available: http://openscience.us/repo/defect/
[54] 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.
[55] D. J. Hand, *Construction and Assessment of Classification Rules*. Chichester, U.K.: Wiley, 1997.
[56] S. Dowdy, S. Wearden, and D. Chilko, *Statistics for Research*, 3rd ed. Chichester, U.K.: Wiley, 2004.
[57] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *Proc. 10th Int. Workshop Artif. Intell. Stat.*, 2005, pp. 57–64.
[58] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proc. 20th Int. Conf. Mach. Learn. (ICML)*, 2003, pp. 912–919.

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 University, 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 Jiaotong 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 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. and M.Tech. degrees in electronics and communication engineering from the BMS Engineering College, Bengaluru, India, in 1995 and 2005, respectively. 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/C++ and MATLAB platform. She is currently with Professional Engineers Ontario, Canada, as an Engineering Intern.