Rough Noise-Filtered Easy Ensemble for Software Fault Prediction

Saman Riaz[1],Ali Arshad[1], Licheng Jiao[2]

[1]School of Computer Science and Technology and School of International Education, Xidian University, Xi'an 710071, China
[2]Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China, International Research Center of Intelligent Perception and Computation and International Joint Collaboration Laboratory of Intelligent Perception and Computation

Correspondence: samanriaz@hotmail.com

Abstract: Software fault prediction is the very consequent research topic for software quality assurance. Data driven approaches provide robust mechanisms to deal with software fault prediction. However, the prediction performance of the model highly depends on the quality of dataset. Many software datasets suffers from the problem of class imbalance. In this regard, under-sampling is a popular data pre-processing method in dealing with class imbalance problem, Easy Ensemble (EE) present a robust approach to achieve a high classification rate and address the biasness towards majority class samples. However, imbalance class is not the only issue that harms performance of classifiers. Some noisy examples and irrelevant features may additionally reduce the rate of predictive accuracy of the classifier. In this paper, we proposed two-stage data pre-processing which incorporates feature selection and a new Rough set Easy Ensemble scheme. In feature selection stage, we eliminate the irrelevant features by feature ranking algorithm. In the second stage of a new Rough set Easy Ensemble by incorporating Rough K nearest neighbor rule filter (RK) afore executing Easy Ensemble (EE), named RKEE for short. RK can remove noisy examples from both minority and majority class. Experimental evaluation on real-world software projects, such as NASA and Eclipse dataset, is performed in order to demonstrate the effectiveness of our proposed approach. Furthermore, this paper comprehensively investigates the influencing factor in our approach. Such as, the impact of Rough set theory on noise-filter, the relationship between model performance and imbalance ratio etc. comprehensive experiments indicate that the proposed approach shows outstanding performance with significance in terms of area-under-the-curve (AUC).

Keywords: Software Fault Prediction, Data Preprocessing, Feature Selection, Rough Set Theory, Class Imbalance, Noise Filter, Easy Ensemble.

1 Introduction

Software fault prediction (SFP) is one of the most common hot research topic in experimental software engineering. The complexity and size of software are rapidly incrementing day by day one possible way to handle this issue by a fixate on elimination of noisy and redundant feature during pre-processing data in early phases of software fault prediction. The classification model is utilized to predict the fault modules and non-fault modules from data on previous versions of the software. To enhance the quality of software assurance much attention is on the process of testing and withal on pre-processing training data [1-6].

To predict fault modules in software data, many researchers have proposed machine learning techniques, including clustering technique, statistical method, and neural network technique. To the best of our knowledge, very few have worked on preprocessing data [4], [7-9]. The key conception of our proposed approach is the prosperous classification results highly depends on software datasets and quality of software dataset can be ameliorated by pre-processing data [9-10] by eliminating the irrelevant features and noisy data.

In last few years, researchers have focused on the quality of software data set, which vigorously affect the performance of fault prediction. Issues concerned in software dataset quality include biased dataset [2], [8], outlier/ noise [3], [4] class imbalance [11-13], [1], [9], [6] and a large dimension of features [5], [14]. In our paper, we focus on three aspects (high feature dimension, outlier, and imbalance class) to enhance the quality of software datasets.

However, some features are irrelevant and even redundant, these features will affect the generalization performance of software fault prediction [15]. Feature selection is the process of selecting the subset of features from the original dataset by removing irrelevant features [5]. Feature selection technique has been proved effective in reducing dimensionality, decrementing time cost, and incrementing result accuracy [16]. In our proposed algorithm, we utilize feature selection to eliminate the feature which is mostly irrelevant to the class. Feature ranking algorithm [17], [18], [19] can be utilized for feature selection according to the ranking of feature with respect to importance (i.e. weights) in differentiating instances of different classes [4-5].

Class imbalance problem is a challenging problem for classification [20-22]. Class imbalance problem occurs when at least one class is significantly fewer than other classes. In our paper, we consider binary class (fault modules (F) and Non-fault modules (NF)), in which number of fault modules are significantly fewer than non-fault modules. As a result, they incline to misclassify fault modules.

To handle this problem, we proposed two steps

1 To eliminate the outliers to reduce the noise
2 Re-sampling to deal with imbalanced classification problem [23].

During the first step of outlier detection, we utilized the amalgamation of the Rough set theory [24] with k-nearest neighbor rule (KNN) [25] which is the most popular learning techniques. Rough set theory deal with overlapping, uncertainty, and vagueness within the data sets and KNN rule is utilized to eliminate the outliers from the dataset on the basis of the ratio of k nearest values. By this amalgamation, we have not only achieved the good accuracy withal reduce the time cost.

Re-sampling is a class of method, which is widely utilized in data pre-processing [9], [27], [28]. Over-sampling and under-sampling transmute the training set by randomly data falling into minority class instance and sampling a smaller majority datasets both are helpful for a class imbalance problem and easy to implement with better results. There are some drawbacks beside the positive achievements. Under-sampling method may ignore some useful data from majority class and over-sampling method cause over fitting. To deal with such drawbacks, researchers have proposed some methods that sample is more involutes complex ways. Easy-Ensemble (EE) [29] is an example, which is the extension of the under-sampling algorithm. Easy-Ensemble independently samples several subsets from majority class and one classifier is built for each subset. All generated classifier are then combined with the final decision by utilizing AdaBoost [30].

Based on the above analysis, we propose the cumulation of feature selection, outlier detection, and re-sampling. This algorithm is effectively eliminating the irrelevant feature, noisy data and balance the class distribution. This approach has not the only potential to achieve the good prediction performance of classification model also reduce the time cost.

This paper is organized as follows. In section 2, we will provide a review of related work, Section 3, deals with the methodology of the algorithm, Section 4, describes experiment and results and section 5, provides with the conclusion.

2 Related work

Software fault prediction is one of the most consequential tasks to predict the fault modules of software modules. Many researchers have focused on classification model [31-38], [9], [39-45] to categorize software into fault and non-fault. These classification models training on the data, which is collected from previous projects, have faulted modules have been identified. Software faults are distributed among imbalanced classes. According to the Parto Principal most software faults are lying into minority class. Thus, to enhance the efficiency and accuracy of prediction performance, data quality is essential for the high classification performance [2], [4], [6], [46].

Data pre-processing is valuable to enhance the software data quality [10], [14], [9] which includes re-sampling and feature selection. Feature selection is utilized to remove the irrelevant feature from the dataset, which will hurt the generalization performance of classification [15]. Feature rank algorithm [12], [5], [10] are utilized for selection on the bases of importance (i.e. weights) in differentiating modules of different classes [4], [5], [17], [18], [47].

Some researcher has proved that not all the samples are utilizable for training datasets to classification [20], [48]. Some samples may be noisy, redundant and not only increase the computational cost also cause the misleading the prediction performance. The Rough set theory is a consequential mathematical tool to deal with the uncertainty and vagueness [9], [10]. The concept of Rough set theory, which is a subset of the universe can be expressed by two sets the lower and upper approximation. Boundary region is different between upper and lower approximations. On the basis of Rough set theory, three-ways decision can be made immediate acceptance (lower approximation) and may or may not be acceptance (boundary region). In our proposed algorithm, the modules in the boundary region can be further discussed the possibility to assign as noisy data.

Qi Kang [13] utilized under-sampling with the KNN noise-Filter during the data pre-processing to deal with class imbalance problem. In this paper, they reduced noise data from minority examples before executing re-sampling. They utilized four popular under-sampling techniques, i.e. under sampling + AdaBoost, under Bagging, RUS Boost, and Easy Ensemble. By the analysis of these results, Easy Ensemble under-sampling techniques achieve high performance. There are many re-sampling methods are proposed to deal with the imbalanced class problem during data pre-processing [23], [27], [49]. Example under-sampling transmutes the training set by samples the majority class, while over-sampling repeating minority class ramply [50]. Both methods are easy and helpful to handle the imbalance training set. However, under-sampling ignores the consequential samples from majority class, while over-sampling is over-fitting due to repetition in the minority class. To overcome these issues, there are also some methods that sample is more complex ways, SMOTE [51] it is over-sampling technique, which creates artificial data on the bases of similarity feature space among minority class. In order to further improvement in SMOTE, many variants have been proposed [49], [52], [53], [54], [55]. SMOTE iteratively partitioning filter [20] it's an over-sampling borderline technique that will remove the noisy data from minority and majority class.

Liu et al [29] proposed the Easy Ensemble [EE] classifier to overcome the deficiency of under-sampling technique [55], which independently samples many subsets from majority class and one classifier are then combined with the final decision by utilizing AdaBoost [30].

In our proposed algorithm we utilized most traditional classification algorithms example decision tree (C4.5), 1-nearest neighbor rule (1-NN) and Naive Bayes (NB) etc [57-62], [63], [64], [28] as a base classifier learner or weak classifier. These classification algorithms are not achieved good results with imbalance dataset. With the combination of the traditional classifier with Easy Ensemble after pre-processing data based on Rough Noise-Filtered to overcome the problems of imbalanced datasets.

Many researchers have utilized the combination of re-sampling techniques with feature selection or noise-filter techniques [11], [13], [20] to handle the imbalance problem. However, the best of our knowledge, there are no attempts the noise filtered under-sampling technique with Rough set theory to improve the data quality of software fault prediction.

In our proposed algorithm, first, we perform feature selection to remove the irrelevant feature and then remove the noisy data by utilizing the combination of rough set theory with KNN rule to redundancy control before applying Easy Ensemble under-sampling to deal with imbalanced datasets and time cost. We evaluate our proposed algorithm by using benchmark NASA & Eclipse dataset which is compared with Area under the curve (AUC).

3 Methodology

In this paper, we pre-processing approach for classification for the imbalanced dataset. Fig 1 gives the framework of our proposed approach. In general, paper consists of two parts. First feature selection by feature ranking algorithm, and second which is the combination of rough set theory and KNN rule afore executing Easy Ensemble (EE). By the above combination, we could get high-quality balanced data for training the classification model to achieve the good prediction accuracy.

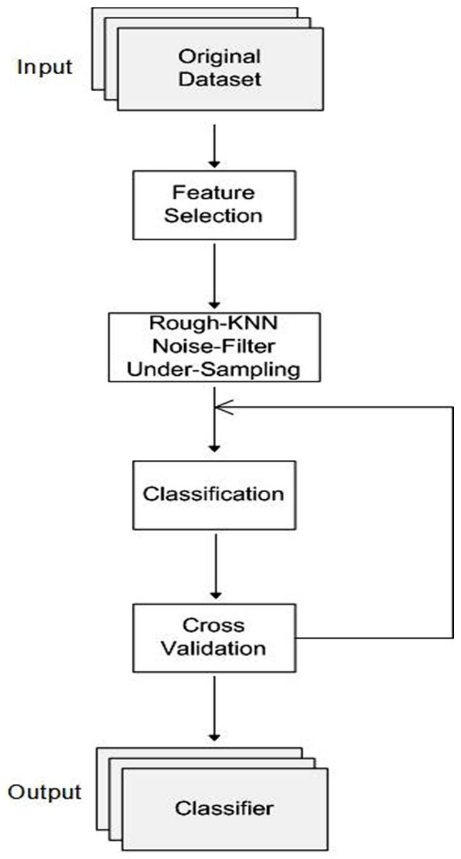

Figure 1. Framework of Proposed Algorithm

3.1. Feature Selection

In the first stage of the framework of our approach, feature selection aims to eliminate the irrelevant features by feature ranking algorithm. The feature is ranked according to the weight of the relevant feature to the class. Generally, weights can be measured by three methods. One is statistics-based method [16], [77-80], second is probability-based method [81], [82], [62], [83–85], [19], and third is instance-based method [19], [86]. In our paper, we select one measuring techniques for feature ranking which has been proven good for imbalanced dataset [87]. The measuring technique is Information Gain (IG) [17].

3.1.1. Information Gain (IG)

Information Gain is the probability-based measuring technique [84]. 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.

$$\mathrm{IG}\,(X/Y) = h(X) - h(X/Y) \tag{1}$$

Rough Noise-Filtered, in equation 2, $h(X)$ is entropy of a discrete random variable X (i.e. the class) which is calculated by

$$h(X) = -\sum_{x\in X} p(x)\log_2 p(x) \tag{2}$$

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

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

3.1.2. Rough- KNN Noise Filter (RK- Filter)

In classification techniques, Noise and outlier in most cases are erroneously created. In our approach, the second stage is shown in fig 2, we proposed a new approach to remove the outliers by the combination of rough set theory and K nearest neighbor rule (KNN). With Easy Ensemble the aims of proposed Rough Noise Filtered technique, rough set theory handle the uncertainty and overlapping dataset, which also reduce the computational time. KNN rule is utilized to remove the outlier on the basis of false k neighbors. Let k is the positive integer, for example, if we have considered the sample x_j from minority sample and all k values by KNN rule are majority class than we can say that x_j is an outlier. Whether the sample is a noisy or not highly depends on the size of k. Suppose we are given dataset S = (U, A), where U are finite sets called universe and A is the set of features, $X \subseteq U$ and $B \subseteq A$. Let us define two operations assigning to every $X \subseteq U$ two sets $\underline{B}(X)$and $\overline{B}(X)$, called the lower and upper approximation of X, respectively and defined as

$$\underline{B}(X) = \cup_{x\in U}\{B(x){:}\, B(x) \subseteq X\} \tag{4}$$

&

$$\overline{B}(X) = \cup_{x\in U}\{B(x){:}\, B(x) \cap X \neq \emptyset\} \tag{5}$$

The set of boundary region is

$$B(X) = \overline{B}(X) - \underline{B}(X) = \{B(x_1), B(x_2), \ldots, B(x_s)\} \tag{6}$$

Where s, is the number of modules in the boundary region, we consider checking the possibility of outliers in the boundary region. Considering the facts that outlier usually locates in boundary by algorithm 1. Considering a given dataset S, Generally, the fault modules (S_F) are in minority and non-fault modules (S_{NF}) are in majority.

Let $S_F \subset S$ & $S_{NF} \subset S$, satisfying $|S_{NF}| > |S_F|$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 May 2018 doi:10.20944/preprints201805.0248.v1

Algorithm 1: RK Noise Filter

Input: B(X), S_F, S_{NF} and k is the number of neighbors, we need to calculate.
Output: New fault set $(S_F^{/})$and non-fault set $(S_{NF}^{/})$.

```
1.  Begin
2.  For j=1:s
      1. Calculate the k nearest neighbors of modules $x_j \in B(X)$from dataset $\{S_F \cup S_{NF}\} - x_j$ through KNN rule.
      2. If $x_j \in S_F$, then count the number of Non-fault modules among k nearest neighbors, otherwise count the number of fault modules.
      3. If $x_j \in S_F$ and all the k nearest neighbor of $x_j$ are non-fault modules then we can consider $x_j$ as the noisy modules and vice versa.
3.  End For
4.  Delete all the noisy modules from $S_F$& $S_{NF}$
5.  Update new $S_F^{/}$ & $S_{NF}^{/}$
```

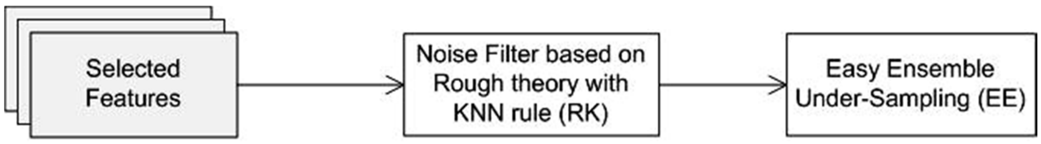
Selected Features
Noise Filter based on Rough theory with KNN rule (RK)
Easy Ensemble Under-Sampling (EE)

Figure 2. Rough Noise-Filtered Easy Ensemble (RKEE)

3.1.3. Easy Ensemble - Rough- KNN Filter (RKEE)

In this paper, we proposed RKEE which is a Rough Noise-Filtered Easy Ensemble technique. It cannot only remove noises from majority class but also from the minority class. RKEE Filter removes noises in dataset before EE [65] in algorithm 2 gives its procedure.

Algorithm 2: RKEE

Input: B(X), S_F, S_{NF}, T and Q
Output: Ensemble model H

```
1.  Begin
2.  Using algorithm 1, RK Noise Filter to remove noisy modules from the dataset and obtain new $S_F^{/}$ & $S_{NF}^{/}$
3.  For $i = 1{:}T$
4.  Randomly sample a subset $S_{NF,i}^{/}$ from $S_{NF}^{/}$ and satisfying $|S_{NF,i}^{/}| = |S_{NF}^{/}|$
5.  Learn the classifier $H_i$ using$S_{NF,i}^{/}$ and $S_F^{/}$ through AdaBoost, $H_i$ is a strong classifier with Q weak Classifiers $h_{ij}$ and $\propto_{ij}$ is the weight of $h_{ij}$. The ensembles threshold is $\Theta_i$ . $H_i(x) = sgn\left(\sum_{j=1}^{Q} \propto_{ij} h_{ij}(x) - \Theta_i\right)$
6.  End For
7.  Output $H(x) = sgn\left(\sum_{i=1}^{T}\sum_{j=1}^{Q} \propto_{ij} h_{ij}(x) - \sum_{i=1}^{T}\Theta_i\right)$
```

4 Experiments

In this paper, we design simulation experiments to evaluate the effectiveness of our proposed approach X-All algorithms. X-All is the combination of feature selection technique (IG) and Rough KNN noise filtered Easy Ensemble with X weak classifiers. First, we introduce the benchmark datasets, which is collected from real-world Software projects, namely the NASA Software Project and Eclipse Project.

4.1. Data Preparation

For illustrating the feasibleness and effectiveness of our proposed algorithm, we evaluate our methods on ten NASA dataset (CM1, MC2, MW1, KC1, KC3, KC4, PC1, PC3, PC4, and PC5) [31], [34], [36], [4] and three Eclipse dataset (Eclipse 2.0, Eclipse 2.1, and Eclipse 3.0) [32], [66-69], which are commonly used for the software fault prediction. The NASA software dataset asre obtained from publicly available MDP (Metrics Data Program) repository [70], while Eclipse dataset are collected from Promise data repository [71].

Table 1. Details of Selected Benchmark Datasets

Dataset	Size	Dimension	$\|S_{NF}\|/\|S_F\|$	Ratio
CM1	505	37	457/48	9.5
MC2	161	39	109/52	2.112
MW1	403	37	372/31	12
KC1	145	86	85/60	1.42
KC3	458	39	415/43	9.65
KC4	125	14	64/61	1.05
PC1	1107	37	1031/76	13.57
PC3	1563	37	1403/160	8.77
PC4	1458	37	1280/178	7.19
PC5	17186	38	16670/516	32.31
Eclipse 2.0	6729	155	5754/975	5.9
Eclipse 2.1	7888	155	7034/854	8.24
Eclipse 3.0	10593	155	9025/1568	5.76

See specific information of experiment dataset in table 1. In this table, size is the number of examples in each dataset. Dimension represent the number of features in each dataset, all dataset have binary class (fault and Non-fault) problems. Fault class is used as the minority class $|S_F|$ and and Non-Fault class is used as majority class $|S_{NF}|$. Ratio represents the imbalanced ratio which is equal to $|S_{NF}|/|S_F|$.

4.2. Performance Measure

Traditionally, prediction accuracy (ACC) is used to evaluate the performance of classification for balanced dataset, but for imbalanced dataset may be misleading. Therefore, in our experiment, AUC (Area under the curve of Receiver operating characteristic (ROC)) [72] is used to measure the classification performance which has been proved to be a reliable evaluation matric for imbalanced classification problems [73]. ROC curves [74] make use of the proportion of two single-column-based evaluation matrices, namely true positive rate and false positive rate obtained by a classifier. To furthermore describe measures T_{tp} as true positive rate and T_{fp} as false positive rate.

$$T_{tp} = \frac{TP}{TP+FN} \tag{7}$$

$$T_{fp} = \frac{FP}{FP+TN} \tag{8}$$

AUC has been widely used to evaluate the performance of classifier for software fault prediction, also can provide a general indication of the predictive potential of the classifier [34], [75]. Due to the lower variance of AUC, AUC performance measure is more reliable than other evaluation matrices such as precision, recall or F-Measure [28].

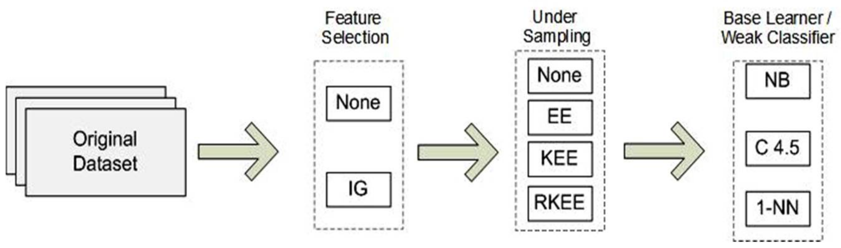

Figure 3. Different Schemes for Experimental Analysis

4.3. Experimental Design

In this paper, we design experiments to demonstrate the effectiveness of our proposed algorithm. In fig 3, we have six different schemes are used for evaluation (X, X-IG, X-EE, X-KEE, X-RKEE, and X-All). X is a learning scheme to train base classifiers, we implement three different base classifier models, which are commonly used in software fault prediction, namely 1-nearest neighbor rule (1-NN) [64], decision tree (C4.5) [63], and Naïve Bayes (NB) [28]. All these three algorithms are implemented based on WEKA 3.5.5 [76], which is the most famous library of machine learning algorithms.

Table 2. Setting of Size K for KNN Noise-Filter with or without Rough Set Theory

Dataset	$\mathbf{k_1}$	$\mathbf{k_2}$
CM1	20	28
MC2	7	12
MW1	12	15
KC1	9	15
KC3	12	20
KC4	6	10
PC1	8	14
PC3	28	35
PC4	16	28
PC5	5	8
Eclipse 2.0	48	73
Eclipse 2.1	32	66
Eclipse 3.0	8	14

Note: k_1 is the size of k without rough set theory and k_2 is the size of k with rough set theory

X-IG is the scheme to train base classifier after feature selection by feature ranking using information Gain (IG) [17]. X-EE is an Easy Ensembling in which X-classifier is used as a weak classifier and then all classifiers are combined with the final decision by utilizing AdaBoost [30]. X-KEE is a Noise-Filtered Easy Ensemble scheme, in which K-nearest neighbor rule (KNN) is applied to minority class for noise-filter [13]. X-RKEE is our core part of the algorithm in which K-nearest neighbor rule (KNN)is applied on the minority and majority classes to remove the noise by using rough set theory to handle the uncertainty and overlapping. X-All is our proposed algorithm, in which all combination of feature selection and Rough Noise-Filtered Easy Ensemble are used for software fault prediction. All experiment results are average over 10 x 10 fold cross-validation. AUC under ROC curve is used to evaluate the performance and Wilcoxon's test is used to analysis of six schemes. We use SPSS software to run Wilcoxon's test. Details of parameter sets are given as follows.

1. NB, C4.5 and1-NN are used to train base classifiers and weak classifier for Easy Ensemble.
2. T = 4 and Q = 10 for Easy Ensemble.
3. We test the performance of K-noise-filter with or without apply rough set theory, when k = 1,2,….,50, which is shown in table 2, best choice for k value is selected on the bases of maximum accuracy.

Any sample from minority and majority class is an outlier is highly based on the size of k.

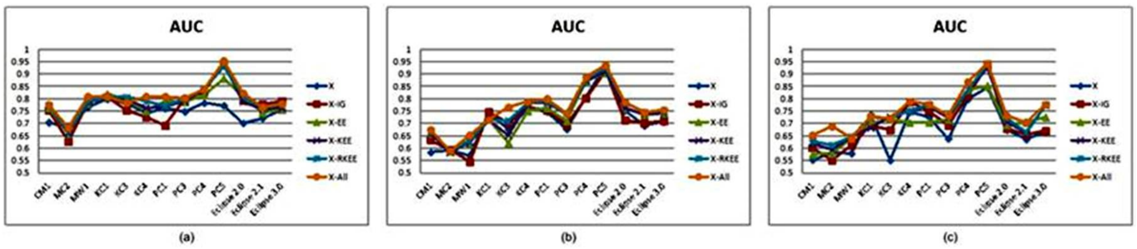

Figure 4. Comparison of Performance of Different Classifier (a) NB (b) C4.5 (c) 1-NN

Table 3. AUC of NB before and after using different combination of feature selection, and under-sampling

Dataset	X	X-IG	X-EE	X-KEE	X-RKEE	X-All
CM1	0.704	0.753	0.763	0.772	0.772	**0.776**
MC2	0.683	0.626	0.657	0.661	0.668	**0.683**
MW1	0.765	0.783	0.780	0.785	0.790	**0.809**
KC1	0.799	0.807	0.820	0.808	0.805	**0.809**
KC3	0.781	0.753	0.797	0.800	**0.807**	0.784
KC4	0.742	0.725	0.751	0.764	0.790	**0.809**
PC1	0.761	0.691	0.790	0.768	0.772	**0.809**
PC3	0.748	0.799	0.788	0.791	0.800	**0.803**
PC4	0.783	0.830	0.816	0.835	0.838	**0.839**
PC5	0.773	0.943	0.883	0.943	0.932	**0.954**
Eclipse 2.0	0.701	0.794	0.810	0.788	0.813	**0.823**
Eclipse 2.1	0.721	**0.778**	0.745	0.758	0.764	0.766
Eclipse 3.0	0.757	**0.791**	0.756	0.762	0.779	0.780
Avg	0.748	0.775	0.781	0.787	0.795	**0.803**

Table 4. AUC of C4.5 before and after using different combination of feature selection, and under-sampling

Dataset	X	X-IG	X-EE	X-KEE	X-RKEE	X-All
CM1	0.583	0.632	0.658	0.658	0.669	**0.674**
MC2	0.590	0.594	0.587	0.582	0.588	**0.589**
MW1	0.570	0.545	0.620	0.623	0.630	**0.652**
KC1	0.741	**0.746**	0.715	0.714	0.715	0.715
KC3	0.676	0.702	0.620	0.651	0.710	**0.765**
KC4	0.764	0.765	0.750	0.783	0.783	**0.789**
PC1	0.748	0.752	0.761	0.783	0.789	**0.800**
PC3	0.677	0.694	0.710	0.738	0.739	**0.741**
PC4	0.808	0.802	0.873	0.874	0.880	**0.889**
PC5	0.933	0.913	0.903	0.911	0.924	**0.936**
Eclipse 2.0	0.752	0.714	0.761	0.762	0.784	**0.786**
Eclipse 2.1	0.692	0.700	0.734	0.739	**0.747**	0.745
Eclipse 3.0	0.706	0.710	0.740	0.742	0.751	**0.755**
Avg	0.711	0.713	0.726	0.735	0.747	**0.757**

Table 5. AUC of 1-NN before and after using different combination of feature selection, and under-sampling

Dataset	X	X-IG	X-EE	X-KEE	X-RKEE	X-All
CM1	0.552	0.601	0.575	0.619	0.628	**0.652**
MC2	0.580	0.550	0.578	0.594	0.613	**0.688**
MW1	0.578	0.612	0.639	0.645	**0.646**	0.639
KC1	**0.735**	0.691	0.701	0.682	0.727	0.729
KC3	0.550	0.673	0.717	**0.728**	0.721	0.721
KC4	0.747	0.788	0.703	0.749	0.752	**0.789**
PC1	0.727	0.751	0.705	0.770	0.775	**0.777**
PC3	0.638	0.692	0.717	0.723	0.731	**0.732**
PC4	0.798	0.815	0.836	0.837	0.837	**0.869**
PC5	0.846	0.937	0.853	0.924	0.942	**0.943**
Eclipse 2.0	0.674	0.679	0.694	0.706	0.724	**0.731**
Eclipse 2.1	0.636	0.656	0.704	0.661	0.665	**0.705**
Eclipse 3.0	0.664	0.669	0.726	0.774	0.776	**0.777**
Avg	0.671	0.701	0.704	0.724	0.734	**0.750**

4.4. Result and Analysis

In this section, table 3, 4, and 5 shows classification results, when base learning schemes are NB, C4.5, and 1-NN respectively. Every table consists of the list of six schemes, every result shows the maximum AUC measure obtained by varing the values of k ranging from 1 to 50 which is the average of 10 x 10 fold cross-validation. Bold values are the best performances in each row of the tables. We can analysis from fig 4, our approach is almost always better than other schemes at all imbalanced ratios. We show Wilcoxon's test results, make the algorithm comparison and perform the analysis on the effectiveness of noise-filter, impact analysis of the combination of feature selection with rough KNN noise-filter and relationship between performance and imbalanced ratio.

Table 6. Wilcoxon's Test Results for the Comparison of X-All (R^{+}) versus X (R^{-}) considering the AUC results

Methods	Draw	R^{+}	R^{-}	$P_{wilcoxon}$
NB-All Vs NB	1	78	0	**0.0022**
C4.5-All Vs C4.5	0	86	5	**0.0047**
1-NN-All Vs 1-NN	0	90	1	**0.0019**

4.4.1. Analysis of X-All verses X

The AUC results for all X-All and X algorithms are shown in table 3, 4, and 5 with base learning algorithm (NB, C4.5, and 1-NN). The best case of each dataset are highlighted in bold. X-All obtains better AUC results compare to X for all base learning s (NB, C4.5, and 1-NN), except NB on MC2 dataset, which has equal performance of software fault prediction , C4.5 on MC2 and KC1 dataset and 1-NN on KC1 dataset respectively. The table 6 shows the Wilcoxon's test result for the comparision of X-All verses X in term of AUC. Column draw is the number of equal results cases. $R^{+}$and $R^{-}$are the sum of ranks $P_{wilcoxon}$ is the p-value of Wilcoxon's test. If $P_{wilcoxon} < 0.05$, it mean the comparisionis significantly different which is in bold font. Therefore, from table 3, 4, and 5, X-All obtain better results , it can also be varified by the Wilcoxon's test result in table 6. Considering all the base learning algorithms in table 6, X-All obtain always better results except NB-All vs NB. All p values less than 0.05 are also indicate that X-All are significantly improve AUC on all X. In summery, the proposed X-All algorithm can effectively improve algorithm performance.

Table 7. Wilcoxon's Test Results for the Comparison of X-KEE (R^{+}) versus X-EE (R^{-}) considering the AUC results

Methods	Draw	R^{+}	R^{-}	$P_{wilcoxon}$
NB-KEE Vs NB-EE	0	76	15	0.0332
C4.5-KEE Vs C4.5-EE	1	78	0	0.0022
1-NN-KEE Vs 1-NN-EE	0	83	8	0.0009

4.4.2. The effectiveness of Noise-Filter through KNN rule

In this subsection, we investigate the performance of software fault prediction with and without Noise removal KNN rule. For this investigation, we will analysis on X-EE versus X-KEE. From table 3, 4, and 5, we can conclude that the performance of EE is improved after noise removed with all base learning algorithms (NB, C4.5 and 1-NN). A Noise-Filered Easy Ensembe through KNN rule [13] is applied only on minority class. However, some times a noise-filter is not good for calssification, it remove useful samples might make a wrong decision. For example, in table 3 the dataset KC1 and PC1 have decreased their performance after noise is removed, futhermore the dataset (CM1, MC2, and KC1) in table 4 and the dataset (KC1 and Eclipse 2.1) in table 5 have not shown good performance after noise-filter. The overall analysis of the effect of Noise-Filter can be seen in table 7 by Wilcoxon's test results, X-KEE show the better prediction performance than X-EE on mostly dataset with (NB, C4.5, and 1-NN) base learner. The all p-values of Wilcoxon's test are also shows that X-KEE can significantly improve AUC.

Table 8. Wilcoxon's Test Results for the Comparison of X-RKEE (R^+) versus X-KEE (R^-) considering the AUC results

Methods	Draw	R^+	R^-	$P_{wilcoxon}$
NB-RKEE Vs NB-KEE	1	67.5	10.5	0.0250
C4.5-RKEE Vs C4.5-KEE	1	78	0	0.0022
1-NN-RKEE Vs 1-NN-KEE	1	72	6	0.0096

4.4.3. Impact of Rough set theory with Noise-Filter

We investigate whether Rough set theory has advantage over Noise-Filter for prediction performance, which is the core part of our algorithm. Outlier is selected on the basis of k value. If the all k nearest neighbor values belong from different classes from the selected sample, selected sample is noisy. Generally, the size of k is smaller for the sample can be classifed as a noise, but if k is large then it will be a good choice. We can see in table 2, the rough set theory highly impacts on the size k. The value of k_2(with rough set theory) is much larger than k_1 (without rough set theory). The large value of k is highly impacted on the classification rate. We conclude results from AUC in table 3, 4, and 5, X-RKEE can further improve prediction performance over X-KEE. X-RKEE is the scheme of noise-filter after apply Rough theory. In this scheme, we invesigate the noise in both classes (Fault and non-Fault). The main advantage of rough theory to handle uncertanity and overlapping problems. Table 8, show the summerized results of Wilcoxon's test on X-RKEE versus X-KEE. We can conclude the result of Wilcoxon's test of AUC, X-RKEE shows better results in most cases. Average eleven out of thirteen dataset have improved there prediction performance. The p-value of Wilcoxon's test 0.0252, 0.0022 and 0.0096 are less than 0.05, which show that X-RKEE are significantly improved X-KEE. After analysis of results we can say that rough set theory with noise-filter improve the prediction performance.

Table 9. Wilcoxon's Test Results for the Comparison of X-All (R^+) versus X-RKEE (R^-) considering the AUC results

Methods	Draw	R^+	R^-	$P_{wilcoxon}$
NB-All Vs NB- RKEE	0	79	12	0.0193
C4.5-All Vs C4.5- RKEE	1	75	3	0.0048
1-NN-All Vs 1-NN- RKEE	1	71.5	6.5	0.0108

4.4.4. The impact of the combination of feature selection with Rough Noise-Filter Easy Ensemble

The combination of feature selection with Rough Noise-Filtered is the pre-processing step for software fault prediction. The prediction preforms highly depend on the quality dataset and dataset can make qualitative by adopting the good pre-processing technique. For this investigation, we compare our approach with or without feature selecting by feature ranking technique (Information Gain (IG)) [17] In table 3, 4, and 5, the column X-All and X-RKEE shows the AUC result with base learners (NB, C4.5, and 1-NN) with and without feature selection respectively. By the analysis of AUC result, we can see that our approach performs better with feature selection technique.on most of the dataset except KC3 with NB base learner, KC1 and Eclipse 2.1 with C4.5 and MW1 with the 1-NN base learner. In table 9, Wilcoxon's test results show that the combination of our approach with feature selection perform significantly better. The p-values of Wilcoxon's test are also much smaller than 0.05, which is also the evidence of better performance of our approach with feature selection.

Table 10. Gain X-All over X, Considering average AUC on (NB, C4.5, and 1-NN)

Dataset	Ratio	AUC		Gain
		X	X-All	
CM1	9.5	0.613	0.701	14.36%
MC2	2.1	0.618	0.653	5.66%
MW1	12	0.638	0.729	14.26%
KC1	1.42	0.758	0.751	-0.92%
KC3	9.65	0.669	0.757	13.15%
KC4	1.05	0.751	0.796	5.99%
PC1	13.57	0.745	0.795	6.7%
PC3	8.77	0.688	0.759	10.32%
PC4	7.19	0.796	0.866	8.79%
PC5	32.31	0.851	0.944	10.93%
Eclipse 2.0	5.9	0.709	0.783	10.44%
Eclipse 2.1	8.24	0.683	0.739	8.20%
Eclipse 3.0	5.76	0.709	0.771	8.74%
Win/Draw/Loss		**12 / 0 / 1**		

4.4.5. Relationship between the performance and imbalanced ratio

In this section, we investigate the relationship between the performance and imbalanced ratio. Table 10, shows the summaries result of AUC for X and X-All respectively on all base learners for each dataset. We can conclude on the basis of table 10. X-All performs better over X on twelve out of thirteen datasets in term of AUC. Prediction performance is reduced only at KC1 datasets with ratio -0.92%. In this paper, dataset with the maximum imbalanced ratio is 32.31 are used for analysis of our proposed approach and perform better with all imbalanced ratio. In our further plan, we will investigate the performance of software fault prediction of our proposed algorithm with ratio 100:1.

5 Conclusion

In this paper, we proposed a novel Rough Noise-Filtered Easy Ensemble for software fault prediction, which incorporates feature selection to enhance the quality of software dataset by removing irrelevent features. Three supervised data classification algorithms (NB, C4.5, 1-NN) are utilized as based classifier in Easy Ensemble and AdaBoost is the fined classifier. The Rough Noise-Filtered is the combination of Rough set theory and K-NN rule, which removed the noisy samples from minority and majority class before applying the Easy Ensemble.

We systematically designed the experiment based on the binary imbalanced class of ten NASA and three Eclipse datasets. AUC measure is used to evaluate the performance of our approach and Wilcoxon's test is used for statistical significancy analysis. The experimental results and significance analysis over thirteen datasets led to the same conclusion. First, Rough Noise-Filtered Easy Ensemble gets better performance. Second, Rough set theory implementation with noise-filter can significantly improve the results by handing the uncertainity and overlapping problem. The results demonstrate the potential of our approach of Rough Noise-Filtered Easy Ensemble with feature selection enhancing the prediction performance.

In our future work, we plan to extend our approach with deep learning classification algorithm and experiment with highly imbalanced data with multi-imbalanced classes, especially those with ratio higher than100:1.

Acknowledgements: We are grateful to Promise Data Repository online, which are used to evaluate our proposed approach. This work was supported in part by the National Basic Research Program (973 Program) of China (No. 2013CB329402), the National Natural Science Foundation of China (No. 61573267, 61473215, 61571342, 61572383, 61501353, 61502369, 61271302, 61272282, 61202176), the Fund for Foreign Scholars in University Research and Teaching Programs (the 111 Project) (No. B07048), the Major Research Plan of the National Natural Science Foundation of China (No. 91438201 and 91438103

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. Software Eng.*, pp. 414–423, 2014.
[2]. C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. T. Devanbu, "Fair and balanced? Bias in bug-fix datasets," in *Proc. Joint Meeting on Foundations of Software Eng.*, pp. 121–130, ACM, 2009.
[3]. S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with noise in defect prediction," in *Proc. IEEE Int. Conf. Software Eng.*, pp. 481–490, 2011.
[4]. M. J. 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, 2013.
[5]. 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. Applications*, pp. 135–140, 2010.
[6]. S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Trans. Rel.*, vol. 62, no. 2, pp. 434–443, 2013.
[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," in *IEEE Transactions on Reliability*, vol. 65, no. 1, pp. 38-53, March 2016.
[8]. K. Herzig, S. Just, and A. Zeller, "It's not a bug, it's a feature: How misclassification impacts bug prediction," in *Proc. IEEE Int. Conf. Software Eng.*, pp. 392–401, 2013.
[9]. H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, 2009.
[10]. 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.
[11]. T. Y. Liu, "EasyEnsemble and Feature Selection for Imbalance Data Sets," *2009 International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing*, Shanghai, pp. 517-520, 2009.
[12]. Ilnaz Jamali, Mohammad Bazmara and Shahram Jafari, "Feature Selection in Imbalance data sets," in IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 2, May 2012.
[13]. Q. Kang, X. Chen, S. Li and M. Zhou, "A Noise-Filtered Under-Sampling Scheme for Imbalanced Classification," in *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4263-4274, Dec. 2017.
[14]. S. Shivaji, E. J. W. , Jr., 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, 2013.
[15]. L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," Journal of Machine Learning Research 2004, vol. 5, pp. 1205–1224, oct 2004.
[16]. I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," Journal of Machine Learning Research, vol. 3, pp. 1157–1182, 2003.
[17]. 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, 2013.
[18]. J. J. Peterson, "Regression analysis of count data," *Technometrics*, vol. 41, no. 4, pp. 371–371, 1999.
[19]. I. Kononenko, "Estimating attributes: Analysis and extensions of relief," in *Proc. Eur. Conf.Mach. Learn.*, pp. 171–182, 1994.
[20]. J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera, "SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering," *Inf. Sci.*, vol. 291, pp. 184–203, Jan. 2015.
[21]. X. Zhang and B.-G. Hu, "A new strategy of cost-free learning in the class imbalance problem," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 2872–2885, Dec. 2014.

[22]. X.-Y. Liu and Z.-H. Zhou, "The influence of class imbalance on costsensitive learning: An empirical study," in *Proc. 6th IEEE Int. Conf. Data Min.*, Hong Kong, pp. 970–974, 2006.
[23]. M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, Boosting-, and hybrid-based appro"aches," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 4, pp. 463–484, Jul. 2012.
[24]. Z. Pawlak, "Rough sets",Int.J.Comput.Inf.Sci.11(5), 341–356, 1982.
[25]. R. Duda, P. Hart, "Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[26]. M. Lin, K. Tang, and X. Yao, "Dynamic sampling approach to training neural networks for multiclass imbalance classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 4, pp. 647–660, Apr. 2013.
[27]. U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 368–386, Jun. 2013.
[28]. X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," In Proceedings of International Conference on Data Mining, pp. 965–969, 2006.
[29]. E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," Machine Learning, vol. 36, no. 2, pp. 105–139, 1999.
[30]. 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, 2007.
[31]. S. Kim, E. J. W. Jr., and Y. Zhang, "Classifying software changes: Clean or buggy?," *IEEE Trans. Softw. Eng.*, vol. 34, no. 2, pp. 181–196, 2008.
[32]. 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. Software Eng.*, pp.181–190, 2008.
[33]. 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, 2008.
[34]. Y. Liu, T. M. Khoshgoftaar, and N. Seliya, "Evolutionary optimization of software quality modeling with multiple repositories," *IEEE Trans.Softw. Eng.*, vol. 36, no. 6, pp. 852–864, 2010.
[35]. Q. Song, Z. Jia, M. J. Shepperd, S. Ying, and J. Liu, "A general software defect-proneness prediction framework," *IEEE Trans. Softw. Eng.*, vol. 37, no. 3, pp. 356–370, 2011.
[36]. P. S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based k-means clustering algorithm," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 6, pp. 1146–1150, 2012.
[37]. J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *Proc. IEEE Int. Conf. Software Eng.*, pp. 382–391, 2013.
[38]. A. Monden, T. Hayashi, S. Shinoda, K. Shirai, J. Yoshida, M. Barker, and K. Ichi Matsumoto, "Assessing the cost effectiveness of fault prediction in acceptance testing," *IEEE Trans. Softw. Eng.*, vol. 39, no. 10, pp. 1345–1357, 2013.
[39]. T. Jiang, L. Tan, and S. Kim, "Personalized defect prediction," in *Proc.Int. Conf. Automated Software Eng.*, pp. 279–289, 2013.
[40]. F. Rahman and P. T. Devanbu, "How, and why, process metrics are better," in *Proc. IEEE Int. Conf. Software Eng.*, pp. 432–441, 2013.
[41]. C. Lewis, Z. Lin, C. Sadowski, X. Zhu, R. Ou, and E. J. W. , Jr., "Does bug prediction support human developers? Findings from a Google case study," in *Proc. IEEE Int. Conf. Software Eng.*, pp. 372–381, 2013.
[42]. F. Zhang, A. Mockus, I. Keivanloo, and Y. Zou, "Towards building a universal defect prediction model," in *Proc. Working Conf. Mining Software Repositories*, pp. 182–191, 2014.
[43]. F. Rahman, S. Khatri, E. T. Barr, and P. T. Devanbu, "Comparing static bug finders and statistical prediction," in *Proc. IEEE Int. Conf. Software Eng.*, pp. 424–434, 2014.
[44]. J. Chen, S. Liu, W. Liu, X. Chen, Q. Gu, and D. Chen, "A two-stage data preprocessing approach for software fault prediction," in *Proc. Int. Conf. Software Security and Reliability*, pp. 20–29, 2014.
[45]. T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1276–1304, 2012.
[46]. [C. Catal, "Software fault prediction: A literature review and current trends," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 4626–4636, 2011.
[47]. J. V. Hulse, T. M. Khoshgoftaar, and A. Napolitano, "A novel noise filtering algorithm for imbalanced data," in *Proc. 9th IEEE Int. Conf. Mach. Learn. Appl.*, Washington, DC, USA, pp. 9–14, 2010.

[48]. T. Maciejewski and J. Stefanowski, "Local neighbourhood extension of SMOTE for mining imbalanced data," in *Proc. IEEE Symp. Comput. Intell. Data Min.*, Paris, France, pp. 104–111, 2011.

[49]. Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," IEEE Trans Knowledge and Data Engineering, vol. 18, no. 1, pp. 63–77, 2006.

[50]. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.

[51]. H. Han, W. Wang, and B. Mao, "Borderline-SMOTE: A new oversampling method in imbalanced data sets learning," in *Proc. Int. Conf. Intell. Comput.* (LNCS 3644), Hefei, China, pp. 878–887, 2005.

[52]. C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-Level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *Proc. 13th Pac. Asia Conf. Adv. Knowl. Disc. Data Min. (PAKDD)*, pp. 475–482, 2009.

[53]. J. Pengfei, Z. Chunkai, and H. Zhenyu, "A new sampling approach for classification of imbalanced data sets with high density," in *Proc. Int. Conf. Big Data Smart Comput.*, Bangkok, Thailand, pp. 217–222, 2014.

[54]. H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE World Congr. Comput. Intell.*, Hong Kong, pp. 1322–1328, 2008.

[55]. G. M. Weiss, "Mining with rarity-problems and solutions: A unifying framework," SIGKDD Explorations, vol. 6, no. 1, pp. 7–19, 2006.

[56]. D. Codetta-Raiteri and L. Portinale, "Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft", *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 1, pp. 13–24, Jan. 2015.

[57]. A. Cano, A. Zafra, and S. Ventura, "Weighted data gravitation classification for standard and imbalanced data," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1672–1687, Dec. 2013.

[58]. Q. Kang, B. Huang, and M. C. Zhou, "Dynamic behavior of artificial Hodgkin–Huxley neuron model subject to additive noise," *IEEE Trans Cybern.*, vol. 46, no. 9, pp. 2083–2093, Sep. 2016.

[59]. K. Nag and N. R. Pal, "A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 499–510, Feb. 2016.

[60]. Q. Kang, J. B. Wang, M. C. Zhou, and A. C. Ammari, "Centralized charging strategy and scheduling algorithm for electric vehicles under a battery swapping scenario," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 3, pp. 659–669, Mar. 2016.

[61]. Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser, "SVMs modeling for highly imbalanced classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 1, pp. 281–288, Feb. 2009.

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

[63]. Q. Song, J. Ni, and G. Wang, "A fast clustering-based feature subset selection algorithm for high-dimensional data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 1–14, 2013.

[64]. I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco, CA, USA: Morgan Kaufmann, 2005.

[65]. T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," in *Proc. Int. Workshop on Predictor Models in Software Eng.*, pp. 9–9, 2007.

[66]. T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: A large scale experiment on data vs. domain vs. process," in *Proc. ACM Joint Meeting on Foundations of Software Eng.*, pp. 91–100, 2009.

[67]. S. Krishnan, C. Strasburg, R. R. Lutz, K. Goseva-Popstojanova, and K. S. Dorman, "Predicting failure-proneness in an evolving software product line," *Inf. Softw. Technol.*, vol. 55, no. 8, pp. 1479–1495, 2013.

[68]. H. Zhang and S. Cheung, "A cost-effectiveness criterion for applying software defect prediction models," in *Proc. ACM Joint Meeting on Foundations of Software Eng.*, pp. 643–646, 2013.

[69]. PROMISE Software Engineering Repository.
http://promise.site.uottawa.ca/SERepository

[70]. Promise Data Repository.
http://promisedata.org/repository

[71]. D. J. Hand, "Construction and Assessment of Classification Rules. John Wiley and Sons", Chichester, 1997.

[72]. J.-H. Xue and P. Hall, "Why does rebalancing class-unbalanced data improve AUC for linear discriminant analysis?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 5, pp. 1109–1112, May 2015.

[73]. T. Fawcett, "ROC graphs: Notes and practical considerations for researchers," *Mach. Learn.*, vol. 31, no. 1, pp. 1–38, 2004.

[74]. Y. Jiang, J. Lin, B. Cukic, and T. Menzies, "Variance analysis in software fault prediction models," in *Proc. Int. Symp. Software Reliability Eng.*, pp. 99–108, 2009.
[75]. Witten I, Frank E, et al, "Weka 3: data mining software in java". University of Waikato, Hamilton, New Zealand, 2007 http://www.cs.waikato.ac.nz/*ml/
[76]. H. Liu and R. Setiono, "Chi2: Feature selection and discretization of numeric attributes," In ICTAI, pages 388–391, 1995.
[77]. R. L. Plackett, Karl pearson and the chi-squared test, "International Statistical Review/Revue Internationale de Statistique," pages 59–72, 1983.
[78]. S. Fong, J. Liang, R. Wong, and M. Ghanavati, "A novel feature selection by clustering coefficients of variations," *In Digital Information Management (ICDIM), 2014 Ninth International Conference*, pages 205–213. IEEE, 2014.
[79]. S. Fong, J. Liang, and Y. Zhuang, "Improving classification accuracy using fuzzy clustering coefficients of variations (fccv) feature selection algorithm", *In Computational Intelligence and Informatics (CINTI), 2014 IEEE 15th International Symposium* on, pages 147–151. IEEE, 2014.
[80]. A. Ahmad and L. Dey, "A feature selection technique for classificatory analysis," *Pattern Recognition Letters*, 26(1):43–56, 2005.
[81]. T. M. Cover and J. A. Thomas, "Elements of information theory," John Wiley & Sons, 2012
[82]. Z. Xu, J. Liu, Z. Yang, G. An, and X. Jia, "The impact of feature selection on defect prediction performance: An empirical comparison," In Software Reliability Engineering (ISSRE), IEEE 27th International Symposium on, pages 309–320. IEEE, 2016.
[83]. M. D. Buhmann, "Radial basis functions," Acta Numerica 2000, 9:1–38, 2000.
[84]. A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," Artificial intelligence, 97(1):245–271, 1997.
[85]. L. Yu and H. Liu. "Feature selection for high-dimensional data: A fast correlation-based filter solution," In ICML, volume 3, pages 856–863, 2003.
[86]. M. Hall, G. Holmes, et al. "Benchmarking attribute selection techniques for discrete class data mining," Knowledge and Data Engineering, IEEE Transactions on, 15(6):1437–1447, 2003.
[87]. B. Ghotra, S. McIntosh and A. E. Hassan, "A Large-Scale Study of the Impact of Feature Selection Techniques on Defect Classification Models," 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), Buenos Aires, pp. 146-157, 2017