https://doi.org/10.1016/j.ins.2018.04.068

A Hybrid Evolutionary Preprocessing Method for Imbalanced Datasets

Ginny Y. Wong*, Frank H.F. Leung*, Sai-Ho Ling**

Abstract

Imbalanced datasets are commonly encountered in real-world classification problems. Many machine learning algorithms are originally designed for well-balanced datasets, therefore re-sampling has become an important step to pre-process imbalanced data. This aims to balance the datasets by increasing the samples of the smaller class or decreasing the samples of the larger class, which are known as over-sampling and under-sampling, respectively. In this paper, a sampling strategy that is based on both over-sampling and under-sampling is proposed, in which the new samples of the smaller class are created based on fuzzy logic. Improvement of the datasets is done by the evolutionary computational method of Cross-generational elitist selection, Heterogeneous recombination and Cataclysmic mu-

*Centre for Signal Processing, Dept. of Electronic and Information Engg., Hong Kong Polytechnic University, Hung Hom, Hong Kong.

**Centre for Health Technologies, Faculty of Engineering and Information Technology, University of Technology Sydney, NSW, Australia.

Email addresses: ginnyyk.wong@connect.polyu.hk (Ginny Y. Wong), frank-h-f.leung@polyu.edu.hk (Frank H.F. Leung), steve.ling@uts.edu.au (Sai-Ho Ling)

tation (CHC) that under-samples both the minority and majority samples. Consequently, a hybrid preprocessing method is proposed to re-sample imbalanced datasets. The evaluation is done by applying the Support Vector Machine (SVM), C4.5 decision tree and nearest neighbor rule to train a classification model from the re-sampled training sets. From the experimental results, it can be seen that our proposed method improves both the $F - measure$ and AUC. The over-sampling rate and complexity of the classification model are also compared. Our proposed method is found to be superior to all other methods under comparison and it is more robust in different classifiers.

1. Introduction

The classification of imbalanced datasets has recently been a popular topic [22] and [27]. Most machine learning tools, such as neural networks and support vector machines (SVMs), were originally designed for well-balanced datasets. Therefore, if the dataset is imbalanced, the performance of the classifier can be poor. The reason for this is apparent. For example, considering a dataset with 99% of data from class A and only 1% of data from class B, then the accuracy is 99% if the classifier ignores the data from class B and labels the whole dataset as class A. It is already very hard to achieve an accuracy above 99% by using most of the learning algorithms. However, the minority class of datasets is usually more important and meaningful. For example, in a medical problem, there are much

fewer samples of people with a particular disease than those of healthy people. If a classifier is needed to label whether some people are infected or not, then the minority class (i.e. people with a particular disease) is the class of interest.

Problems with imbalanced datasets can easily be found in the real world, such as intrusion detection [9], speech recognition [26], identification of power distribution fault causes [41], and bioinformatics problems [16]. There are two main approaches to solve problems caused by imbalanced datasets: the first is the data level approach and the second is the algorithm level approach. The data level approaches [3], [8], [18], and [28] include balancing the class distribution by over-sampling the minority class or under-sampling the majority class. The algorithm level approaches improve the existing machine learning methods by adjusting the probabilistic estimate [38], modifying the cost per class [32], adding some penalty constants [25], or learning from one class instead of two classes [35] and [30].

Many experiments show that re-sampling is a good data level approach to handle imbalanced data; see, for example, [12], [15], and [42]. Moreover, it is more flexible because it does not depend on the chosen classifier. Therefore, we will focus on re-sampling in this paper. There are three main types of strategies for re-sampling data. The first is over-sampling, which can be done randomly or by the Synthetic Minority Over-sampling Technique (SMOTE) [8]. The second is under-sampling, which includes Tomek links [37] and the Neighborhood Cleaning Rule (NCL) [24]. The last is the hybrid method, which combines the two previous

methods (over-sampling and under-sampling methods).

The importance of designing sampling strategies has been discussed in [31], which may affect the successful learning of different classes. Hybrid re-sampling methods are reported to have the advantage of treating datasets with a high imbalanced ratio, see [3] and [6]. Although some hybrid methods have been proposed to reduce the over-generalization problem from over-sampling methods, most of these methods are based on SMOTE and the results may be limited by the synthetic samples of SMOTE, see [3], [34], and [40] . Therefore, a hybrid re-sampling method is proposed in this paper. Fuzzy logic, which is a useful tool to treat imbalanced datasets [12], is used to over-sample the minority class samples instead of SMOTE. A fuzzy rule base is formed based on the samples of the minority class. A rule is then selected randomly with reference to the effectiveness of each rule. The selected rule is used as the criteria to generate a new sample of the minority class. These steps will repeat until the majority class and minority class are the same size.

A large over-sampled training dataset will increase the complexity of the classification model and decrease the efficiency of the learning algorithm. It will also easily cause over-generalization, especially for some noisy datasets. This happens because the decision boundary could become narrow or the overlapping area between the majority class and minority class could become large after over-sampling. Therefore, an evolutionary algorithm (EA) is applied to both the syn-

thetic samples and majority samples to under-sample the dataset. The chosen EA is the Cross-generational elitist selection, Heterogeneous recombination and Cataclysmic mutation (CHC) algorithm [11], which is able to select the most representative instances among the many algorithms studied in [5].

We will carry out experiments to compare our proposed method with three SMOTE-extended over-sampling methods, four hybrid re-sampling methods, and one under-sampling method, which are: SMOTE, Safe-Level-SMOTE [4], Adaptive Synthetic Sampling [21], SMOTE+Tomek Links [3], SMOTE+Rough Set [34], SMOTE+CHC (sCHC) [40], agglomerative hierarchical clustering [10], and EUSCHC [14]. A total of 44 imbalanced datasets from the UCI Repository [2] are used in the experiments. The SVM [7], C4.5 decision tree [33], and nearest neighbor rule (1NN) are used as tools to reach a classification model for each re-sampled dataset and evaluate each re-sampling method. The evaluation measures are based on the $F - measure$ and the area under the receiver operating characteristic curve (AUC). Although there are many hybrid pre-processing methods, only some of them are similar to our method, and consider and focus on the data size. In this paper, CHC is used to reduce the data size and achieve a good performance. Additionally, the proposed method enhances the performance in the over-sampling stage by taking advantage of the fuzzy rule base.

The rest of this paper is organized as follows. In Section 2, some preprocessing methods and CHC are reviewed. Section 3 presents the details of the proposed

re-sampling strategy and the evaluation method. To show the effectiveness of our proposed approach, the comparisons with other methods and the results are discussed in Section 4. We will draw a conclusion in Section 5.

2. Previous Work

This section describes some previous works that have used re-sampling methods, which will then be compared with our proposed method in the experiments. The concepts of the CHC will also be discussed.

2.1. Re-sampling Methods

As discussed in the previous section, there are three main strategies for re-sampling data, which will be described in more detail in the following subsections.

2.1.1. Over-sampling Methods

Some instances are produced for the minority class to balance the class distribution. The simplest is a non-heuristic method (random over-sampling) that replicates samples of the original minority class to generate the new instances. This method easily causes over-fitting because the new instances copy exactly from the original minority class. SMOTE [8] is a well-known method that creates the new instances by interpolating several minority samples that join together. This method makes use of each minority class sample and inserts synthetic samples along the line segments, joining any/all of the k minority class nearest neighbors

to over-sample the minority class. An example is shown in Fig. 1. Five nearest neighbors are used, where x_i is a selected sample of minority class, x_{i1} to x_{i5} are the five nearest neighbors of x_i, and s_1 to s_5 are the synthetic samples created by interpolation. If the degree of over-sampling required is 300%, then three synthetic examples are selected randomly from s_1 to s_5.

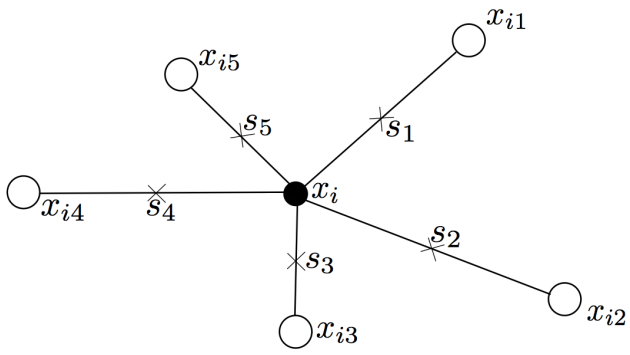


Figure 1: Example of SMOTE with five nearest neighbors.

Because the synthetic samples provide a less specific and larger decision region, the over-fitting problem can be reduced. However, this method may introduce more minority synthetic samples in the area of majority class, where the minority class is very sparse with respect to the majority class. This causes the problem of over-generalization, which means that the decision boundary is very narrow or there is a large overlapping area between the majority class and minority class. Therefore, some methods have been developed based on SMOTE to overcome this limitation, such as Borderline-SMOTE (sBorder) [19], Adaptive Synthetic Sampling (ADASYN) [21], Safe-Level-SMOTE (sSafe) [4], and SPI-

DERS [29].

2.1.2. Under-sampling Methods

Some instances of majority class are eliminated to balance the class distribution. The simplest is random under-sampling (RUS), which aims to balance the datasets by randomly removing samples of the majority class. However, this method may easily remove some useful data. The other representative methods include: (i) condensed nearest neighbor rule (CNN) [20], which eliminates the majority class samples that are distant from the decision border; (ii) Tomek links (TL) [37], which edits out noisy and borderline majority class samples; (iii) one-sided selection (OSS) [23], which is an integrated method of TL and CNN; and, (iv) the neighborhood cleaning rule (NCL) [24], which is based on the Wilson's Edited Nearest Neighbor Rule (ENN) [39] to remove the majority class samples that lead to misclassification.

2.1.3. Hybrid Methods

Although both over-sampling and under-sampling can balance the class distribution, they do have drawbacks, such as over-generalization and the removal of useful data. Therefore, some hybrid methods have been developed to combine SMOTE and under-sampling as a data cleaning method to reduce the problem. Example hybrid methods include SMOTE+Tomek links (sTL), which uses TL to remove samples of both classes to increase the area of decision border,

and SMOTE+ENN (sENN) [3], which uses ENN to remove the samples that are misclassified by their nearest neighbors. Rough set theory (sRST) [34] and evolutionary algorithm (sCHC) [40] have also been applied on SMOTE to select the samples and increase the accuracy of the classification.

Most of these hybrid methods make use of SMOTE to perform over-sampling. Clustering techniques have also been developed to perform under-sampling and over-sampling, such as agglomerative hierarchical clustering (AHC) [10].

2.2. CHC [11]

CHC is a kind of EA that combines a selection strategy with a highly disruptive recombination operator. To avoid premature convergence and maintain diversity, incest prevention and cataclysmic mutation are introduced. The CHC process can be described as follows. First, a population set of chromosomes P is created. Each chromosome $p_i = (p_{i1}, p_{i2}, \ldots, p_{in})$ is an n-dimensional vector, which is a set of genes, where p_{ij} is the jth gene value ($j = 1, 2, \ldots, n$) of the ith chromosome in the population ($i = 1, 2, \ldots, m$), m is the population size, and n is the number of genes. Second, the chromosomes are evaluated by a defined fitness function. The form of the fitness function depends on the application. Third, an intermediate population set of chromosomes C, which is of the same size as P, is generated by copying all of the members of P in a random order.

A uniform crossover (HUX) operator is then applied on C to form C'. HUX

exchanges half of the genes randomly between the chromosomes one by one to form C'. CHC also uses an additional method for incest prevention. Before applying HUX to the chromosomes, the Hamming distance between them is calculated. If half of that distance is larger than a difference threshold d, then HUX is applied; otherwise these two chromosomes are deleted from C. Therefore, the size of C' may be smaller than that of P or C. The initial threshold d is set at $n/4$. After C' has formed, it is evaluated by the fitness function and an elitist selection is taken. Only the best chromosomes from both P and C' are selected to form the offspring population in the next generation. If the offspring population is the same as P, then the difference threshold d is decreased by one.

CHC is different from the traditional genetic algorithm because mutation is not performed at the recombination stage. CHC performs partial reinitialization (divergence) when the search becomes trapped (i.e., the difference threshold d becomes zero and no new offspring population is formed for several generations). The population is reinitialized, based on the best chromosome, by changing the elements' values randomly with a user-defined divergence rate D_{rate}. For example, if D_{rate} equals to 0.35, the values of 35% elements will be changed randomly. The search is then resumed with a new difference threshold $d = D_{rate} * (1 - D_{rate}) * n$. This process is called cataclysmic mutation.

CHC has shown the ability to select the most representative instances among the other algorithms studied in [5]. Therefore, it is chosen to improve the outcome

of over-sampling in this paper.

3. Methodology

In this section, the proposed hybrid preprocessing method and the evaluation methods used in this paper are discussed. The proposed method has two stages. The minority samples of the training sets are first over-sampled based on fuzzy logic to form a fuzzy rule base (FRB). To improve the performance, CHC is then implemented to reduce both the synthetic samples and majority samples.

3.1. Fuzzy Rule Base (FRB)

In this paper, let the $positive$ class be the minority class and only λ training samples (X_α) of positive class are considered, where $X_\alpha = (x_{\alpha 1}, \ldots, x_{\alpha\gamma})$ is an γ-dimensional vector, $\alpha = 1, 2, \ldots, \lambda$ and $x_{\alpha\beta}$ is the βth attribute value ($\beta = 1, 2, \ldots, \gamma$) of the αth training sample. The θth fuzzy if-then rule is written as follows:

$$\begin{aligned} &\text{Rule } \theta : \text{IF } z_1 \text{ is } A_1^\theta \text{ AND } \ldots \text{ AND } z_\gamma \text{ is } A_\gamma^\theta \\ &\qquad\qquad \text{THEN class = positive with } w_\theta \end{aligned} \tag{1}$$

where A_β^θ is a fuzzy term of the θth rule corresponding to the attribute z_β, $\beta = (1, 2, \ldots, \gamma)$ and $z = (z_1, z_2, \ldots, z_\gamma)$ is a γ-dimensional attribute vector, and w_θ is the rule weight. The regular triangular membership functions are used for the fuzzy terms. In this paper, the fuzzy terms A_β^θ are derived based on the samples

of positive class. The minimum and maximum values of each attribute are first found. The fuzzy terms are the triangular membership functions within the range of each attribute. The fuzzy terms also depend on the number of labels. Because regular triangular membership functions are used, the fuzzy terms are distributed evenly within the range of each attribute.

The fuzzy rules are generated based on the samples of positive class. For each sample, the label with the highest membership value is selected to form the corresponding rule for each attribute. The maximum number of rules depends on the number of labels and attributes.

The rule weight w_θ is used to reflect the degree of matching of each fuzzy rule over all the positive samples, so that the importance of each rule can be evaluated. First, the fuzzy value of each sample is calculated. The fuzzy value of X_α for the θth fuzzy rule is then defined as follows:

$$\mu_{A^\theta}(X_\alpha) = T(\mu_{A_1^\theta}(x_{\alpha 1}), \ldots, \mu_{A_\gamma^\theta}(x_{\alpha\gamma})), \tag{2}$$

where the product T-norm is used. The rule weight (w_θ) is calculated by adding all the fuzzy values of samples.

$$w_\theta = \sum_{\alpha=1}^{\lambda} (\mu_{A^\theta}(X_\alpha)). \tag{3}$$

After the rule base of the positive class is generated, the rules are randomly drawn based on the rule weight. The rule that has a higher rule weight will have a

higher probability to be chosen. Then, a new sample is generated within the area of the selected rule. These processes are repeated until the number of positive samples is the same as that of the negative samples.

To illustrate this idea more clearly, Fig. 2 shows the distribution of two classes with two attributes as an example of the formulation of fuzzy rules. The x-axis and y-axis govern the values of the two different attributes and regular triangular membership functions with five labels are used. The circle dots correspond to the negative class and the square dots correspond to the positive class. The dashed lines show the minimum or maximum value of the corresponding attribute of the positive samples. Because only the attribute vectors of the positive class are considered to generate fuzzy rules, a total of 10 rules can be formed in this example:

Rule 1: IF z_1 is $A_1^1 = L1\_1$ AND z_2 is $A_2^1 = L2\_4$. THEN class = positive with 0.897

Rule 2: IF z_1 is $A_1^2 = L1\_2$ AND z_2 is $A_2^2 = L2\_3$. THEN class = positive with 1.147

Rule 3: IF z_1 is $A_1^3 = L1\_2$ AND z_2 is $A_2^3 = L2\_4$. THEN class = positive with 1.508

Rule 4: IF z_1 is $A_1^4 = L1\_3$ AND z_2 is $A_2^4 = L2\_3$. THEN class = positive with 1.230

Rule 5: IF z_1 is $A_1^5 = L1\_3$ AND z_2 is $A_2^5 = L2\_4$. THEN class = positive with 2.344

Rule 6: IF z_1 is $A_1^6 = L1\_3$ AND z_2 is $A_2^6 = L2\_5$. THEN class = positive with 1.607

Rule 7: IF z_1 is $A_1^7 = L1\_4$ AND z_2 is $A_2^7 = L2\_1$. THEN class = positive with 0.727

Rule 8: IF z_1 is $A_1^8 = L1\_4$ AND z_2 is $A_2^8 = L2\_4$. THEN class = positive with 1.319

Rule 9: IF z_1 is $A_1^9 = L1\_4$ AND z_2 is $A_2^9 = L2\_5$. THEN class = positive with 1.731

Rule 10: IF z_1 is $A_1^{10} = L1\_5$ AND z_2 is $A_2^{10} = L2\_4$. THEN class = positive with 1.399

where z_1 and z_2 represent Attribute 1 and Attribute 2 for the x-axis and y-axis, respectively, in Fig. 2, $L1\_i$ is the i-th label of z_1 attribute, $L2\_i$ is the i-th label of z_2 attribute. Rule 5 has the highest rule weight and rule 7 has the lowest rule weight in this example.

To generate the synthetic samples, one rule out of these 10 rules is chosen with the probability of selection depending on the rule weight. This rule then sets the criteria of the highest and lowest value of each attribute. The new sample is generated randomly within these criteria. This process is repeated until the number of

the positive class is the same as that of the negative class. Fig. 3 shows the samples' distribution after over-sampling. The triangle dots represent the synthetic samples. It is found that the spread of the synthetic samples is similar to that of the original positive samples (shown as the square dots). The synthetic samples in Fig. 3 are dense in the area of rule 5.

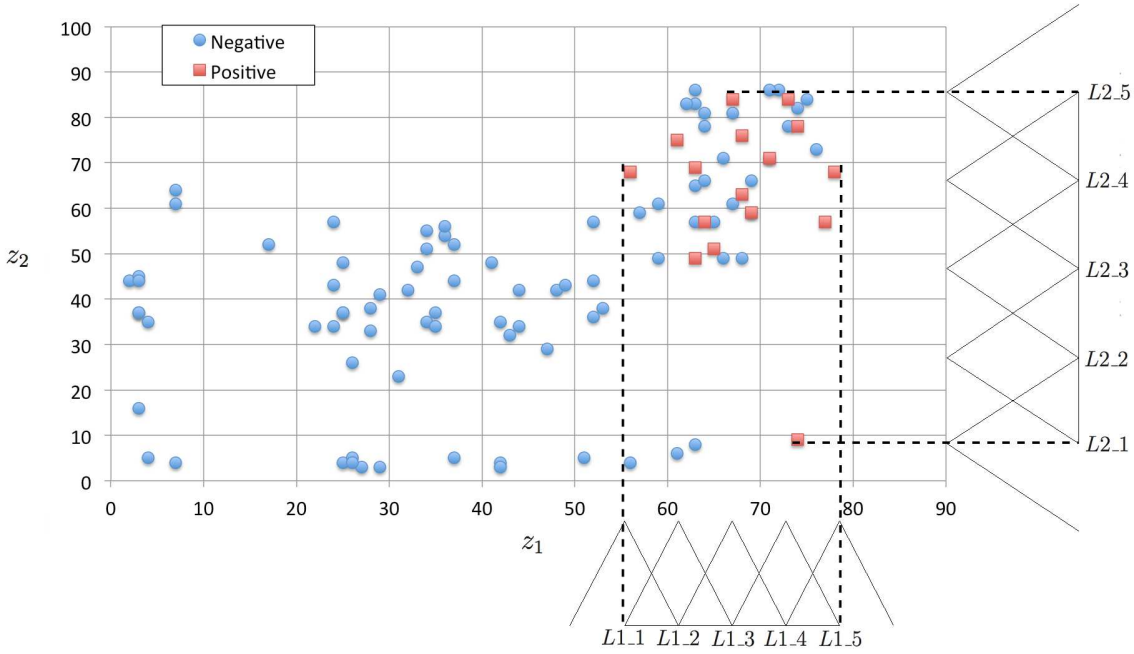


Figure 2: Example of the distribution of imbalanced dataset. The y-axis represents the values of z_2 and x-axis represents the value of z_1.

3.2. Setting the CHC

After over-sampling, the number of minority samples is the same as that of majority samples and CHC is then applied. The representation of each chromosome

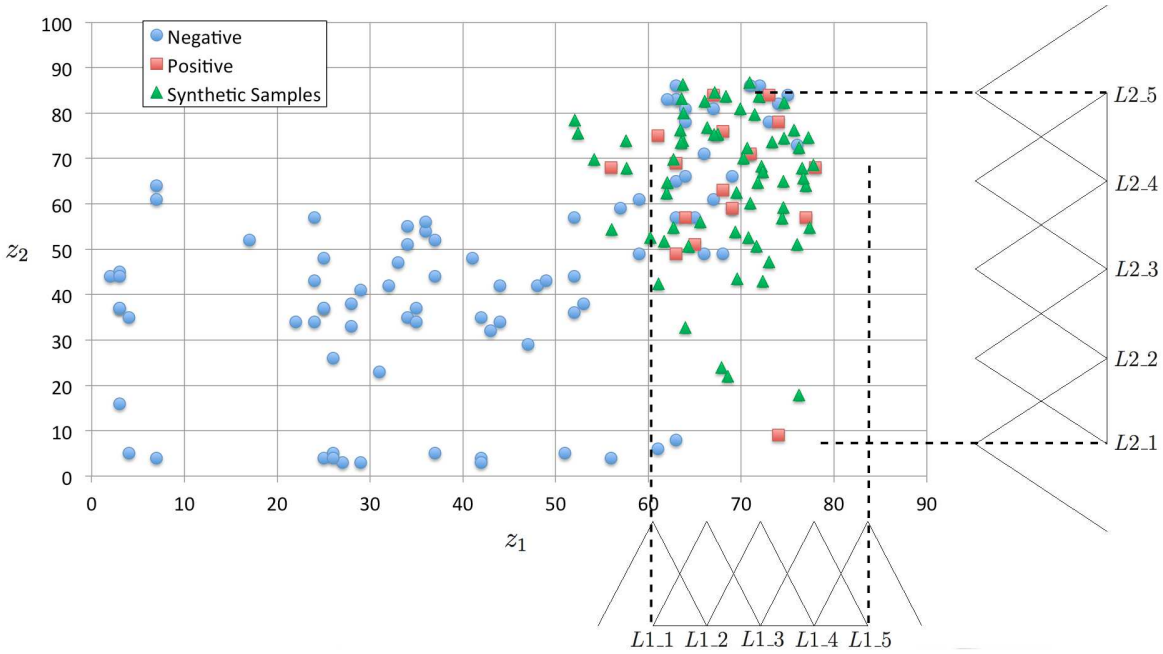


Figure 3: Distribution of the samples after over-sampling. The y-axis represents the values of z_2 and x-axis represents the value of z_1.

and the definition of fitness function need to be addressed before the algorithm is employed. Fig. 4 shows the block diagram of the process of FRB+CHC.

3.2.1. Chromosome Representation

CHC is used to reduce the synthetic samples and also the majority class samples. Therefore, the chromosomes are used to represent subsets of these samples. This can be carried out by a binary representation. Each chromosome is an n-dimensional vector. In this section, n is the number of synthetic samples plus majority class samples. Each vector element shows whether or not the corresponding sample exists in the subset of the training set. Therefore, there are two possible

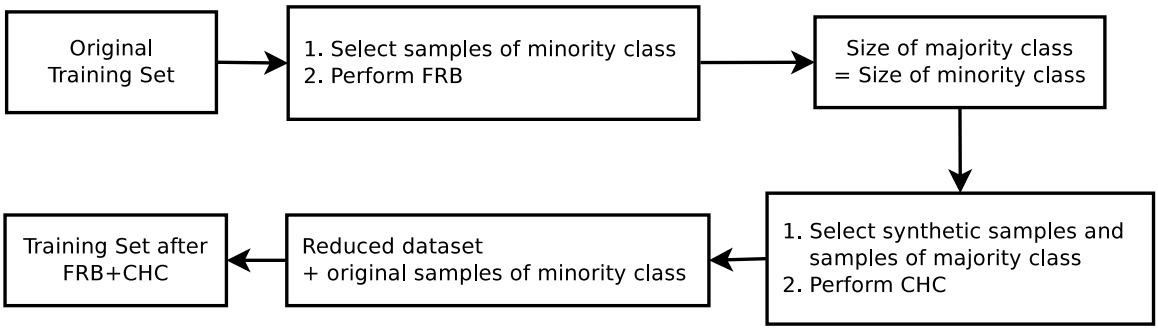


Figure 4: Block diagram of FRB+CHC.

values for each element: 0 and 1. If the value is 1, then the corresponding sample is included in the subset of the training set. If the value is 0, then the sample does not exist in the subset.

3.2.2. Fitness Function

In this study, the k-NN classifier is used as the evaluation method of CHC to obtain the subset with the highest classification rate. Normally, accuracy (i.e. the ratio of correctly classified samples to total number of samples) would be used as the measure of classification rate. However, this may cause difficulty for the imbalanced datasets during testing because the correct classification rate of the majority samples may affect the accuracy more significantly than that of the minority samples. Therefore, some other measures are used in this paper. These measures are commonly employed to analyze problems with imbalanced datasets.

First, precision and recall are introduced [17]. Their definitions are given as

follows:

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

where TP is the number of true positives, FP is the number of false positives and FN is the number of false negatives. A high value of precision indicates that the predicted positive samples are most likely to be relevant. A high value of recall indicates that most of the positive samples can be predicted correctly.

A popular evaluation metric for imbalanced problems is $F - measure$ [17], which is a function of precision and recall. In principle, $F - measure$ represents a harmonic mean between precision and recall. A high value of $F - measure$ means both the precision and recall values are high and do not differ very much. This is an important measure for imbalanced datasets because a high value can imply that the method classifies the positive samples correctly at a high rate with little misclassified negative samples. This is defined as follows:

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6}$$

The area under the receiver operating characteristic curve (AUC) is also commonly used to measure the performance of classification. The AUC measure [13] is the probability of correctly identifying a random sample, and it can be defined

as follows:

$$AUC = \frac{1 + Recall - FP_{rate}}{2} \tag{7}$$

where $Recall$ is defined in (5) and $FP_{rate} = \frac{FP}{FP+TN}$, TN is the number of true negatives. FP_{rate} defines the percentage of true negatives cases misclassified as positives. A high value of AUC implies small values of FN and FP, which indicates that the corresponding classifier is effective.

Because both $F-measure$ and AUC are important measures on imbalanced datasets, a multi-objective fitness function is used here. The chromosome with both higher values of $F-measure$ and AUC obviously has a higher rank. If a chromosome X has a higher value of $F-measure$ ($F_X > F_Y$) and a lower value of AUC ($A_X < A_Y$) than that of chromosome Y, then the difference between the chromosomes' $F-measure$ ($|F_X - F_Y|$) and the difference between the chromosomes' AUC ($|A_X - A_Y|$) will be compared. If $|F_X - F_Y| > |A_X - A_Y|$, then chromosome X will be regarded as the better option; otherwise chromosome Y will be regarded as the better option. This setting is also applied in sCHC for the comparison in Section 4.

3.3. Evaluation

3.3.1. $F-measure$ and AUC measures

To show the performance of our proposed method, $F-measure$ in (6) and AUC in (7) are used. The main drawback of over-sampling or hybrid sampling

methods is that the number of training samples are increased greatly. This may increase the complexity of the learning model. Therefore, the over-sampling rates of different methods are also compared. Define:

$$Rate_{over} = \frac{(N_{sampled} - N_{original})}{N_{original}} * 100\% \tag{8}$$

where $N_{sampled}$ is the number of samples in the re-sampled training set and $N_{original}$ is the number of samples in the original training set. The over-sampling rate in (8) shows the increase rate of the number of the training samples. When a SVM is used to form the classification mode, the increase rate of the support vectors can be used to evaluate the complexity of the learning model. This rate is calculated based on the support vectors generated.

$$Rate_{SV} = \frac{(SV_{sampled} - SV_{original})}{SV_{original}} \tag{9}$$

where $SV_{sampled}$ is the number of support vectors trained by the re-sampled training set and $SV_{original}$ is the number of support vectors trained by the original training set. It should be noted that the CHC fitness evaluation for data size reduction (by k-NN) and the training of the classification model based on the resampled data (by SVM) are two separate processes. K-NN is used in the fitness evaluation because it is simple and has minimal computation effort. SVM is a commonly used method to obtain the classification model.

4. Experimental Study

In this section, we will present the experiments that are carried out to compare our proposed method with other hybrid sampling methods and the CHC under-sampling method. The datasets that we have used can be found in UCI Repository [2].

The experiments involve several different kinds of hybrid methods, including SMOTE, ADASYN, sTL, sSafe, sRST, sCHC, AHC and our proposed method, which is named Fuzzy Rule Base+CHC (FRB+CHC). CHC, which is used as an under-sampling method in [14] (EUSCHC), is also compared in the experiment. To measure the performance of the preprocessing methods, the same learning tool should be applied among all the experiments. In this study, three different tools are used, which are the SVM, 1 Nearest Neighbor (1NN), and C4.5 decision tree. The programs of all testing methods and the learning tools are based on KEEL, which is open source software that is available via the Web [1]. $F - measure$ and AUC are used as measures to analyze the results. The average values of these measures for each method will be calculated. Because the expansion of re-sampled training datasets may increase the computational time and complexity of the classification model, the over-sampling rate and the number of support vectors formed from SVM will also be compared.

4.1. Datasets

To study these methods on different datasets, 44 datasets with different imbalance ratio (IR) are chosen. IR is the ratio of the number of majority class to the number of minority class. Table 1 shows the details of the selected datasets, where the number of samples ($N_{samp.}$), the number of attributes ($N_{attr.}$), the distribution of the minority and majority classes, and IR for each dataset can be found.

Table 1: Details of the Selected Imbalanced Datasets.

Dataset	$N_{samp.}$	$N_{attr.}$	**Min., Maj.(%)**	**IR**
ecoli034vs5	200	7	(10, 90)	9
yeast2vs4	514	8	(9.92, 90.08)	9.08
ecoli067vs35	222	7	(9.91, 90.09)	9.09
ecoli0234vs5	202	7	(9.9, 90.1)	9.1
glass015vs2	172	9	(9.88, 90.12)	9.12
yeast0359vs78	506	8	(9.88, 90.12)	9.12
yeast0256vs3789	1004	8	(9.86, 90.14)	9.14
yeast02579vs368	1004	8	(9.86, 90.14)	9.14
ecoli046vs5	203	6	(9.85, 90.15)	9.15
ecoli01vs235	244	7	(9.83, 90.17)	9.17
ecoli0267vs35	224	7	(9.82, 90.18)	9.18
glass04vs5	92	9	(9.78, 90.22)	9.22
ecoli0346vs5	205	7	(9.76, 90.24)	9.25
ecoli0347vs56	257	7	(9.73, 90.27)	9.28
yeast05679vs4	528	8	(9.66, 90.34)	9.35
vowel0	988	13	(9.01, 90.99)	9.98
ecoli067vs5	220	6	(9.09, 90.91)	10

Dataset	$N_{samp.}$	$N_{attr.}$	**Min., Maj.(%)**	**IR**
glass016vs2	192	9	(8.85, 91.15)	10.29
ecoli0147vs2356	336	7	(8.63, 91.37)	10.59
led7digit02456789vs1	443	7	(8.35, 91.65)	10.97
ecoli01vs5	240	6	(8.33, 91.67)	11
glass06vs5	108	9	(8.33, 91.67)	11
glass0146vs2	205	9	(8.29, 91.71)	11.06
glass2	214	9	(7.94, 92.06)	11.59
ecoli0147vs56	332	6	(7.53, 92.47)	12.28
cleveland0vs4	177	13	(7.34, 92.66)	12.62
ecoli0146vs5	280	6	(7.14, 92.86)	13
shuttlec0vsc4	1829	9	(6.72, 93.28)	13.87
yeast1vs7	459	7	(6.53, 93.47)	14.3
glass4	214	9	(6.07, 93.93)	15.47
ecoli4	336	7	(5.95, 94.05)	15.8
page_blocks13vs4	472	10	(5.93, 94.07)	15.86
abalone918	731	8	(5.65, 94.25)	16.4
glass016vs5	184	9	(4.89, 95.11)	19.44
shuttlec2vsc4	129	9	(4.65, 95.35)	20.5
yeast1458vs7	693	8	(4.33, 95.67)	22.1
glass5	214	9	(4.2, 95.8)	22.78
yeast2vs8	482	8	(4.15, 95.85)	23.1
yeast4	1484	8	(3.43, 96.57)	28.1
yeast1289vs7	947	8	(3.16, 96.84)	30.57
yeast5	1484	8	(2.96, 97.04)	32.73
ecoli0137vs26	281	7	(2.49, 97.51)	39.14
yeast6	1484	8	(2.36, 97.64)	41.4

Dataset	$N_{samp.}$	$N_{attr.}$	Min., Maj.(%)	IR
abalone19	4174	8	(0.77, 99.23)	129.44

4.2. Setup of the Experiment

For over-sampling, the rules of the minority samples are associated with regular triangular membership functions with five fuzzy terms. For CHC, the values of the parameters are:

- Population size: 50.
- Divergence rate: 0.35.
- Threshold decreasing rate: 0.001.
- k of k-NN classifier used as evaluation: 1.
- Number of evaluations: 5,000.

In this paper, SVM, 1NN, and C4.5 are used to weigh the influence of each preprocessing method. For SVM, a radial basis function (RBF) is used as the kernel because a non-linear classification model is needed and RBF is a commonly used kernel to handle this problem. The RBF is defined as follows:

$$RBF = exp(-\frac{1}{\sigma}\|\mathbf{x}_i - \mathbf{x}\|^2) \tag{10}$$

where $\sigma > 0$ is the parameter to determine the width of the radial basis function, which controls the flexibility of the classifier. When σ decreases, the flexibility of

the resulting classifier in fitting the training data increases, and this might easily lead to over-fitting. The value of σ is set as 0.01. The tradeoff between training error and margin of SVM is set as 100. These values are chosen through experiments. For C4.5, the confidence level is set as 0.25, the minimum number of item-sets per leaf is set to 2, and pruning is used as well to obtain the final tree. For 1NN, the Euclidean distance metric is used.

A five-fold cross validation model is used to compare the classification results from different preprocessing methods. Each dataset are first divided into five parts randomly. Four of them are combined to form a training set and the remaining subset forms a testing set. The process is then repeated five times, so that each subset is used once as a testing set. All of the methods involve some random parameters, so five experiments are carried out for each five-fold cross validation model and the average value are calculated as the results; that is, in total, 25 experiments were done.

4.3. Results

4.3.1. $F-measure$ and AUC measures

Tables 2 and 3 show the SVM results on $F-measure$ and AUC for each re-sampling method on the 44 datasets respectively. The results of the original datasets are shown in the second column and the best value for each dataset are highlighted in bold. The last row shows the average value of each sampling

method for the datasets. The performance of the FRB over-sampling method are also included (in the rightmost column) for comparison with FRB+CHC. It can be seen that the average values of $F-measure$ and AUC in both FRB and FRB+CHC are higher than other methods. The performance of sCHC and FRB+CHC are similar. This shows that CHC has good performance as a data cleaning method after over-sampling, especially for the results in $F-measure$. The AUC values of SMOTE, sTL, sSafe, sRST and sCHC are similar since they all use SMOTE to perform over-sampling. ADASYN gets the lowest average values of $F-measure$, which means the precision is low and the difference between precision and recall is large.

In this experiment, the performance of FRB and FRB+CHC is very similar, which shows the advantages of FRB over the other hybrid or over-sampling methods. However, the data size will be very large if only FRB is used as the preprocessing method. FRB+CHC can reduce the data size without a large effect to the performance. Therefore, only FRB+CHC will be considered in the following section.

Table 4 shows the average rankings by means of $F-measure$ and AUC using Friedman's method [36]. The highest value of each dataset is ranked as 1. If a certain method obtains the ranking 3, 6, 2, and 1 on four datasets, then the average ranking is $(3+6+2+1)/4 = 3$. Therefore, a lower average ranking indicates that the corresponding method is better among the other methods. FRB+CHC obtains

the best ranking by AUC and sCHC obtains the best ranking by $F - measure$. Note that the highest average values of AUC or $F - measure$ do not imply the best ranking results because the ranking shows the comparison results among all of the methods of each dataset. For example, EUSCHC has the lowest AUC average values but its ranking is better than ADASYN. Given that EUSCHC is an under-sampling method, it easily ignores some useful samples of the majority class.

Table 2: SVM: Average $F-measure$ of Testing Datasets among Different Sampling Methods.

Dataset	Original	SMOTE	ADASYN	sTL	sSafe	sRST	sCHC	EUSCHC	AHC	FRB+CHC	FRB
ecoli034vs5	0	0.5629	0.2667	0.5901	0.5578	0.5007	0.5054	**0.6591**	0.3111	0.5829	0.6337
yeast2vs4	0.6384	0.6824	0.5446	0.6683	0.6824	0.6787	0.6996	**0.7418**	0.6937	0.7015	0.6971
ecoli067vs35	0.0000	0.4540	0.3975	0.5122	0.4609	0.4447	0.5108	0.4758	0.3692	0.4308	**0.6171**
ecoli0234vs5	0.0000	0.5176	0.2917	0.5240	0.5012	0.4734	0.5577	**0.6682**	0.2667	0.6142	0.6462
glass015vs2	0.0000	0.3094	0.3181	0.3103	0.3301	**0.3419**	0.2137	0.1015	0.2850	0.2049	0.2275
yeast0359vs78	0.3481	0.3541	0.2965	0.3379	0.3580	0.3529	**0.4117**	0.3481	0.3666	0.3470	0.3481
yeast0256vs3789	0.1782	0.5282	0.4391	0.5206	0.5286	0.5325	0.5624	**0.6033**	0.5263	0.5899	0.2589
yeast02579vs368	0.8152	0.7199	0.5213	0.7179	0.7189	0.7201	0.7437	0.7487	0.7264	0.7747	**0.85**
ecoli046vs5	0.0000	0.3901	0.2000	0.3958	0.4084	0.4214	0.3827	**0.6786**	0.0667	0.5225	0.6584
ecoli01vs235	0.0000	0.4325	0.1648	0.4396	0.4352	0.4264	0.4844	**0.5691**	0.1385	0.4224	0.5536
ecoli0267vs35	0.0000	0.3158	0.2269	0.3257	0.2902	0.3253	0.3856	0.4035	0.1469	0.4592	**0.5337**
glass04vs5	**1.0000**	0.8793	0.8679	0.8747	0.9228	0.9209	0.9933	0.7854	**1.0000**	0.9631	0.9655
ecoli0346vs5	0.0000	0.5446	0.3636	0.6397	0.5741	0.5642	0.5985	**0.7382**	0.3404	0.6766	0.6768
ecoli0347vs56	0.0000	0.5743	0.4743	0.5628	0.5576	0.5104	0.5913	**0.6669**	0.1846	0.5176	0.5913
yeast05679vs4	0.0000	0.4327	0.4265	0.4282	0.4333	0.4250	**0.5066**	0.4996	0.4189	0.4786	0.5355
vowel0	**1.0000**	0.9936	0.9796	0.9905	0.9890	0.9816	0.9833	0.9396	**1.0000**	0.9060	0.9387
ecoli067vs5	0.0000	0.3260	0.2973	0.3463	0.3444	0.3225	0.3787	**0.6848**	0.2308	0.6173	0.6873
glass016vs2	0.0000	0.3196	0.3203	0.2686	0.3048	0.2963	0.2102	0.1395	**0.3404**	0.2001	0.2857
ecoli0147vs2356	0.0000	0.4230	0.3014	0.4960	0.4354	0.4435	0.5021	0.2230	0.0500	0.4043	**0.5074**
led7digit02456789vs1	**0.7748**	0.5707	0.6197	0.5226	0.5766	0.5156	0.7308	0.5691	0.5961	0.6746	0.7224
ecoli01vs5	0.0000	0.4138	0.2588	0.4482	0.4103	0.4946	0.4392	0.4140	0.2069	0.6843	**0.7811**
glass06vs5	**1.0000**	0.9057	0.9655	0.8953	0.8857	0.9083	0.9866	0.8654	**1.0000**	0.9783	0.9474
glass0146vs2	0.0000	0.2463	0.2512	0.2247	0.2473	0.2814	0.2823	0.1747	**0.2931**	0.2597	0.2768
glass2	0.0000	0.2477	0.2362	0.2329	0.2478	0.2988	0.2484	0.1131	**0.3233**	0.2019	0.26

Dataset	Original	SMOTE	ADASYN	sTL	sSafe	sRST	sCHC	EUSCHC	AHC	FRB+CHC	FRB
ecoli0147vs56	0.0000	0.5757	0.4148	0.6288	0.6022	0.5103	0.5164	0.6148	0.0571	0.6762	**0.7609**
cleveland0vs4	0.0000	0.1539	0.1556	0.1560	0.1263	0.1600	0.0923	**0.2621**	0.0000	0.1687	0.2030
ecoli0146vs5	0.0000	0.4280	0.1920	0.4112	0.4356	0.4422	0.3762	0.6993	0.3000	0.7456	**0.7758**
shuttlec0vsc4	0.9490	0.9740	0.8937	0.9749	0.9740	**0.9817**	0.9724	0.9707	0.9675	0.7964	0.8763
yeast1vs7	0.0000	0.2926	0.2870	0.2865	0.2939	0.2738	0.3120	0.0000	0.2861	0.3161	**0.3381**
glass4	**0.8560**	0.6633	0.6565	0.6590	0.6613	0.6463	0.8190	0.7164	0.8471	0.7273	0.7197
ecoli4	0.7500	0.6352	0.5082	0.6354	0.6389	0.6491	**0.7931**	0.7372	0.7109	0.7356	0.6617
page_blocks13vs4	0.2270	0.2033	0.1907	0.2010	0.2034	0.1894	**0.3563**	0.0832	0.2270	0.1816	0.1907
abalone918	0.0444	0.4522	0.4172	0.4206	0.4474	0.4570	0.5221	0.2643	0.5303	**0.5732**	0.5561
glass016vs5	0.6650	0.5674	0.6592	0.5601	0.5668	0.6551	0.7548	0.4688	0.7273	**0.7694**	0.6857
shuttlec2vsc4	0.4	0.7152	0.7152	0.7152	0.7152	0.7288	0.6103	0.1593	0.4	0.6126	**0.7395**
yeast1458vs7	0	0.1318	0.1261	0.1260	0.1323	0.1344	**0.1585**	0	0.1398	0.1557	0.1187
glass5	0.7	0.5937	0.4551	0.5495	0.5932	0.4838	0.6583	0.3542	0.7	0.7533	**0.8**
yeast2vs8	0.6967	0.5972	0.2079	0.5905	0.5989	0.5984	**0.7068**	0.6967	0.6570	0.6967	0.6967
yeast4	0	0.2703	0.2464	0.2648	0.2715	0.2711	0.3076	0.0308	0.2714	**0.3533**	0.3398
yeast1289vs7	0	0.1395	0.1363	0.1357	0.1397	0.1308	0.1851	0	0.1488	**0.1967**	0.1776
yeast5	0	0.4843	0.4611	0.4742	0.4818	0.4751	0.5146	**0.5802**	0.5012	0.4476	0.4415
ecoli0137vs26	0	0.3976	0.2400	0.4681	0.4292	0.3636	**0.4306**	0.3158	0.1	0.3465	0.3826
yeast6	0	0.2698	0.2014	0.2606	0.2705	0.2670	**0.3577**	0	0.2756	0.3288	0.2759
abalone19	0	0.0408	0.0406	0.0403	0.0409	**0.0486**	0.0437	0	0.0411	0.0482	0.0445
Mean	0.2510	0.4711	0.3917	0.4734	0.4733	0.4693	0.5090	0.4492	0.4039	0.5179	**0.5451**

Table 3: SVM: Average AUC of Testing Datasets among Different Sampling Methods.

Dataset	Original	SMOTE	ADASYN	sTL	sSafe	sRST	sCHC	EUSCHC	AHC	FRB+CHC	FRB
ecoli034vs5	0.4972	0.7069	0.5889	0.7236	0.7047	0.6799	0.6747	0.8111	0.5972	0.8217	**0.8472**
yeast2vs4	0.7362	0.8924	0.8788	0.8900	**0.8931**	0.8892	0.8656	0.8804	0.8885	0.8757	0.8424
ecoli067vs35	0.5000	0.6860	0.6625	0.7063	0.6790	0.6700	0.6943	0.6675	0.6200	0.7860	**0.8325**
ecoli0234vs5	0.4972	0.6978	0.6140	0.7081	0.6943	0.6820	0.7181	0.8014	0.5917	**0.8289**	0.8112
glass015vs2	0.5000	0.7152	0.7352	0.7284	0.7376	**0.7496**	0.5905	0.4911	0.6484	0.5530	0.575
yeast0359vs78	0.6067	0.7344	0.6936	0.7281	**0.7391**	0.7334	0.7289	0.6067	0.7371	0.6062	0.6067
yeast0256vs3789	0.5486	0.7960	0.7734	0.7972	0.7965	0.7993	0.8038	**0.8064**	0.7918	0.7691	0.5761
yeast02579vs368	0.8695	0.9057	0.8610	0.9085	0.9035	0.9071	0.9041	**0.9135**	0.9052	0.9125	0.9078
ecoli046vs5	0.4973	0.6496	0.5614	0.6488	0.6574	0.6696	0.6395	0.7461	0.5195	0.7880	**0.8427**
ecoli01vs235	0.4955	0.6606	0.5377	0.6628	0.6598	0.6616	0.6758	0.7423	0.5405	0.7866	**0.8659**
ecoli0267vs35	0.5000	0.6073	0.5826	0.6093	0.6020	0.6113	0.6405	0.7035	0.5450	0.8176	**0.8483**
glass04vs5	**1.0000**	0.9754	0.9754	0.9728	0.9842	0.9830	0.9988	0.9570	**1.0000**	0.9732	0.9938
ecoli0346vs5	0.4973	0.6974	0.6115	0.7421	0.7124	0.7127	0.7170	0.7878	0.6169	0.8459	**0.8656**
ecoli0347vs56	0.5000	0.7569	0.7028	0.7594	0.7444	0.7294	0.7511	0.8071	0.5579	0.7888	**0.8310**
yeast05679vs4	0.5000	0.7869	0.7902	0.7862	0.7861	0.7797	**0.7934**	0.7860	0.7754	0.7899	0.7786
vowel0	**1.0000**	0.9993	0.9978	0.9990	0.9988	0.9981	0.9982	0.9933	**1.0000**	0.9892	0.9933
ecoli067vs5	0.5000	0.6103	0.6100	0.6155	0.6175	0.6106	0.6245	0.8000	0.5725	**0.8125**	0.845
glass016vs2	0.5000	**0.7529**	**0.7529**	0.7106	0.7464	0.7322	0.6239	0.5733	0.7517	0.6114	0.6552
ecoli0147vs2356	0.4984	0.6509	0.6154	0.6920	0.6580	0.6629	0.6891	0.6504	0.5102	0.8054	**0.8441**
led7digit02456789vs1	0.8788	0.8819	0.8867	0.8799	0.8856	0.8650	0.8946	**0.9055**	0.8600	0.8844	0.8921
ecoli01vs5	0.4977	0.6602	0.5864	0.6786	0.6566	0.6875	0.6659	0.7091	0.5727	0.8159	**0.8432**
glass06vs5	**1.0000**	0.9774	0.9950	0.9574	0.9629	0.9436	0.9895	0.9397	**1.0000**	0.9840	0.95
glass0146vs2	0.5000	0.6823	0.6849	0.6594	0.6821	0.7142	0.6717	0.5519	**0.7153**	0.6336	0.6808
glass2	0.5000	0.7132	0.6981	0.6938	0.7127	0.7607	0.6648	0.5248	**0.7868**	0.6078	0.6875

Dataset	Original	SMOTE	ADASYN	sTL	sSafe	sRST	sCHC	EUSCHC	AHC	FRB+CHC	FRB
ecoli0147vs56	0.5000	0.7160	0.6352	0.7460	0.7335	0.7053	0.6905	0.7722	0.5167	0.8578	**0.9171**
cleveland0vs4	0.4969	0.5622	0.5575	0.5526	0.5321	0.5421	0.5210	0.5991	0.4811	0.5857	**0.6034**
ecoli0146vs5	0.4981	0.6440	0.5654	0.6394	0.6467	0.6558	0.6260	0.7731	0.5962	0.8371	**0.8442**
shuttlec0vsc4	0.9515	0.9747	0.9872	0.9755	0.9747	0.9845	0.9731	0.9715	0.9749	0.9812	**0.9897**
yeast1vs7	0.5000	0.7583	**0.7744**	0.7632	0.7602	0.7500	0.6777	0.5000	0.7261	0.6932	0.7579
glass4	0.9092	0.9148	0.9176	0.9113	0.9143	0.9163	0.9333	0.9251	**0.9350**	0.9230	0.8942
ecoli4	0.8000	0.9101	0.9149	0.9143	0.9171	0.9426	0.9244	**0.9528**	0.9279	0.9368	0.9231
page_blocks13vs4	0.5700	0.7528	0.7320	0.7493	**0.7531**	0.7298	0.6847	0.5609	0.5689	0.7141	0.732
abalone918	0.5125	0.8961	0.8860	0.8863	0.8939	0.8916	0.8745	0.5792	**0.9144**	0.8597	0.83
glass016vs5	0.8443	0.8856	0.9186	0.8791	0.8853	**0.9221**	0.8979	0.8071	0.8943	0.9186	0.8886
shuttlec2vsc4	0.7	0.9548	0.9548	0.9548	0.9548	**0.9590**	0.9440	0.6957	0.7	0.9493	0.9632
yeast1458vs7	0.5	0.6427	0.6373	0.6396	0.6444	**0.6539**	0.6638	0.5	0.6546	0.5958	0.5954
glass5	0.8451	0.8760	0.8256	0.8807	0.8845	0.8515	0.8515	0.8768	0.8451	**0.8967**	0.8927
yeast2vs8	0.7739	0.7628	0.7242	0.7614	0.7633	0.7770	0.7852	0.7739	**0.8381**	0.7739	0.7739
yeast4	0.5	0.8156	0.8102	**0.8227**	0.8160	0.8124	0.8177	0.5093	0.8127	0.7991	0.7663
yeast1289vs7	0.5	0.7141	0.7145	0.7133	0.7109	0.6968	0.7201	0.5	0.7202	0.6990	**0.7453**
yeast5	0.5	0.9668	0.9635	0.9655	0.9665	0.9655	0.9683	0.7976	**0.9691**	0.9621	0.9611
ecoli0137vs26	0.5	0.7118	0.5927	0.7390	**0.7413**	0.6909	0.7294	0.6427	0.5463	0.6655	0.6945
yeast6	0.5	0.8742	0.8597	0.8716	0.8744	0.8736	0.8735	0.5	0.8761	**0.8880**	**0.8880**
abalone19	0.5	0.7177	0.7170	0.7163	0.7180	**0.7715**	0.7166	0.5	0.6881	0.7016	0.7063
Mean	0.6141	0.7784	0.7519	0.7805	0.7795	0.7801	0.7703	0.7248	0.7339	0.8020	**0.8133**

Table 4: Friedman Rankings of AUC and $F-measure$.

Preprocessing Method	AUC	F-measure
Original	8.841	7.568
SMOTE	4.636	5.341
ADASYN	6.159	7.341
sTL	4.864	5.636
sSafe	4.624	4.932
sRST	4.659	5.318
sCHC	4.886	**3.477**
EUSCHC	6	5.659
AHC	5.773	5.5
FRB+CHC	**4.455**	3.886

Although the hybrid sampling methods can get better results, their main drawback is that the size of the training set is greatly expanded. If the IR of the dataset is large, then the size of the re-sampled training set can be nearly double that of the original. This drawback may increase the computational time and complexity of the learning model. Table 5 shows the over-sampling rates of different methods on each dataset and the mean rate of each method. A negative value means that the size of re-sampled training set is smaller than that of the original. A value greater than 100% means that the size of re-sampled training set is more than two times that of the original set. Both sCHC and FRB+CHC shrink most of the dataset while the over-sampling rates of the other methods are similar. This shows that both sCHC and FRB+CHC can use less training samples to achieve high perfor-

mance. Table 6 shows the details of the re-sampled training sets after applying FRB+CHC. It reveals the decrease rate of the majority class, the increase rate of the minority class, and the updated IR. The IR values of the re-sampled training sets are not always equal to 1 because CHC makes use of a fitness function to select a subset of samples. The range of IR is between 0.9 and 1.5.

Table 7 shows the increase rate of the number of support vectors used to form the classification model. The number of support vectors can reflect the complexity of the classification model formed by SVM. When the number of support vectors is smaller, the classification model is more easily applied. Some negative values can be found because the number of support vectors for the re-sampled dataset is less than that of the original dataset. Both sCHC and FRB+CHC have the smallest increase rate of the number of support vectors, on average. The average number of support vectors are only increased by around 0.776 times and 0.948 times of the original datasets, while most of the other methods have increased by over two times.

The results of sCHC and FRB+CHC are similar. To show the differences between these two methods, Fig. 5 reveals the average AUC results obtained from the training and testing sets (sorted by the nonlinearity of the 1NN classifier). The x-axis shows the selected 44 datasets, the solid lines in the figures represent the average AUC results for the testing set, and the dashed lines represent the average AUC results for the training set. FRB+CHC shows the advantage on relaxing the

Table 5: Over-sampling Rate (%) of Training Sets among Different Sampling Methods.

Dataset	SMOTE	ADASYN	sTL	sSafe	sRST	sCHC	AHC	FRB+CHC
ecoli0347vs56	80.53	80	77.63	80.53	100.77	-5.60	80	-3.06
yeast2vs4	80.16	80.16	76.85	80.16	80.16	-3.13	80.16	-3.90
ecoli067vs35	80.18	80.18	77.14	80.18	94.13	-5.19	80.18	-4.16
ecoli0234vs5	80.20	80.20	77.10	80.20	89.36	-4.46	80.20	-5.50
glass015vs2	80.23	80.23	70.79	80.23	80.23	-2.06	80.23	-2.28
yeast0359vs78	80.24	80.24	71.49	80.24	80.34	-4.90	80.24	-2.14
yeast0256vs3789	80.28	80.28	74.25	80.28	80.73	-5.70	80.28	-0.92
yeast02579vs368	80.28	80.28	76.97	80.28	80.28	-3.46	80.28	-2.52
ecoli046vs5	80.30	80.30	77.09	80.30	112.06	-3.99	80.30	-2.72
ecoli0147vs2356	84.01	80.33	80.02	84.01	119.57	-4.05	80.33	-1.50
ecoli0267vs35	80.36	80.36	76.56	80.36	94.65	-4.03	80.36	-2.62
glass04vs5	80.44	80.44	77.18	80.44	96.76	-4.40	80.44	-2.91
ecoli034vs5	80.20	80.49	77.72	80.20	86.63	-4.43	80.49	-2.73
ecoli0346vs5	80.39	80.54	78.18	80.39	87.98	-3.54	80.54	-3.12
yeast05679vs4	80.68	80.68	74.48	80.68	80.68	-5.71	80.68	-2.48
vowel0	81.78	81.78	81.78	81.78	84.56	-4.82	81.78	-4.29
ecoli067vs5	81.82	81.82	75.68	81.82	87.27	-4.78	81.82	-1.68
glass016vs2	82.29	82.29	73.18	82.29	82.29	-0.78	82.29	-2.11
ecoli0137vs26	90.48	82.74	87.65	90.48	169.85	-1.33	82.74	0.28
led7digit02456789vs1	83.30	83.30	78.39	83.30	94.02	-3.93	83.30	-7.90
ecoli0147vs56	83.97	83.33	79.91	83.97	137.03	-3.39	83.33	-1.73
glass06vs5	83.34	83.34	80.79	83.34	91.22	-2.71	83.34	-1.67
glass0146vs2	83.41	83.41	74.63	83.41	83.41	-0.94	83.41	-1.40
glass2	84.11	84.11	75.93	84.11	84.11	-2.07	84.11	-0.42
ecoli0146vs5	89.30	84.94	86.00	89.30	139.65	-3.34	84.94	-2.73
cleveland0vs4	84.97	84.97	80.35	84.97	205.49	-3.27	84.97	-1.13
ecoli01vs5	92.75	85.71	90.61	92.75	186.47	-3.58	85.71	-1.22
shuttlec0vsc4	86.55	86.55	86.50	86.55	136.58	-3.32	86.55	-3.17
yeast1458vs7	91.81	86.93	87.16	91.81	91.81	-1.13	86.93	-2.51
glass4	87.85	87.85	83.65	87.85	112.84	-2.68	87.85	-0.95
ecoli4	88.10	88.10	86.46	88.10	88.39	-1.59	88.10	-1.40
page_blocks13vs4	88.14	88.14	86.60	88.14	157.10	-2.88	88.14	-0.86
abalone918	88.58	88.58	83.38	88.58	88.58	-1.72	88.58	-1.44
glass016vs5	90.22	90.22	88.45	90.22	94.57	-2.84	90.22	-1.03
shuttlec2vsc4	90.70	90.70	89.92	90.70	113.19	-3.86	90.70	-3.43
yeast1289vs7	92.32	91.34	88.44	92.32	92.32	-2.50	91.34	-2.35
glass5	91.59	91.59	89.37	91.59	92.76	-1.58	91.59	0.12
yeast2vs8	91.70	91.70	89.83	91.70	98.44	-1.86	91.70	-0.46
yeast4	93.13	93.13	90.09	93.13	93.13	-1.69	93.13	-1.16
yeast1vs7	89.62	93.66	85.04	89.62	89.62	-3.39	93.66	-3.31
yeast5	94.07	94.07	92.62	94.07	94.07	-1.66	94.07	-2.09
ecoli01vs235	82.59	95.02	78.95	82.59	106.04	-4.04	95.02	-0.59
yeast6	95.28	95.28	93.36	95.28	95.28	-2.54	95.28	-1.88
abalone19	98.47	98.47	97.32	98.47	98.47	1.04	98.47	-0.31
Mean	85.70	85.40	81.94	85.70	103.47	**-3.13**	85.40	-2.17

Table 6: The Details of the Re-sampled Datasets After FRB+CHC.

Dataset	Decrease Rate of Majority Class	Increase Rate of Minority Class	Updated IR
ecoli034vs5	0.474	3.700	1.010
yeast2vs4	0.469	3.865	0.995
ecoli067vs35	0.500	4.224	0.874
ecoli0234vs5	0.488	3.463	1.048
glass015vs2	0.405	2.989	1.369
yeast0359vs78	0.418	3.995	1.064
yeast0256vs3789	0.452	4.005	1.002
yeast02579vs368	0.456	3.841	1.028
ecoli046vs5	0.469	3.900	1.001
ecoli01vs235	0.445	3.939	1.030
ecoli0267vs35	0.500	3.756	0.969
glass04vs5	0.464	4.325	0.934
ecoli0346vs5	0.476	4.188	0.935
ecoli0347vs56	0.452	3.940	1.032
yeast05679vs4	0.441	3.938	1.064
vowel0	0.514	4.619	0.863
ecoli067vs5	0.461	4.025	1.076
glass016vs2	0.393	3.635	1.357
ecoli0147vs2356	0.424	4.651	1.084
led7digit02456789vs1	0.499	4.603	0.983
ecoli01vs5	0.470	4.975	0.978
glass06vs5	0.482	5.339	0.910
glass0146vs2	0.392	4.568	1.212
glass2	0.449	4.826	1.101
ecoli0147vs56	0.475	5.510	0.992
cleveland0vs4	0.442	6.049	0.984
ecoli0146vs5	0.470	5.550	1.053
shuttlec0vsc4	0.507	6.530	0.908
yeast1vs7	0.420	5.525	1.277
glass4	0.489	6.651	1.038
ecoli4	0.463	7.300	1.024
page_blocks13vs4	0.480	7.236	1.003
abalone918	0.478	7.658	0.994
glass016vs5	0.486	9.296	0.974
shuttlec2vsc4	0.514	9.900	0.936
yeast1458vs7	0.391	7.908	1.513
glass5	0.478	10.321	1.052
yeast2vs8	0.452	10.500	1.105
yeast4	0.468	12.397	1.117
yeast1289vs7	0.418	11.967	1.373
yeast5	0.480	14.768	1.081
ecoli0137vs26	0.467	19.633	1.023
yeast6	0.451	17.693	1.217
abalone19	0.487	61.247	1.067

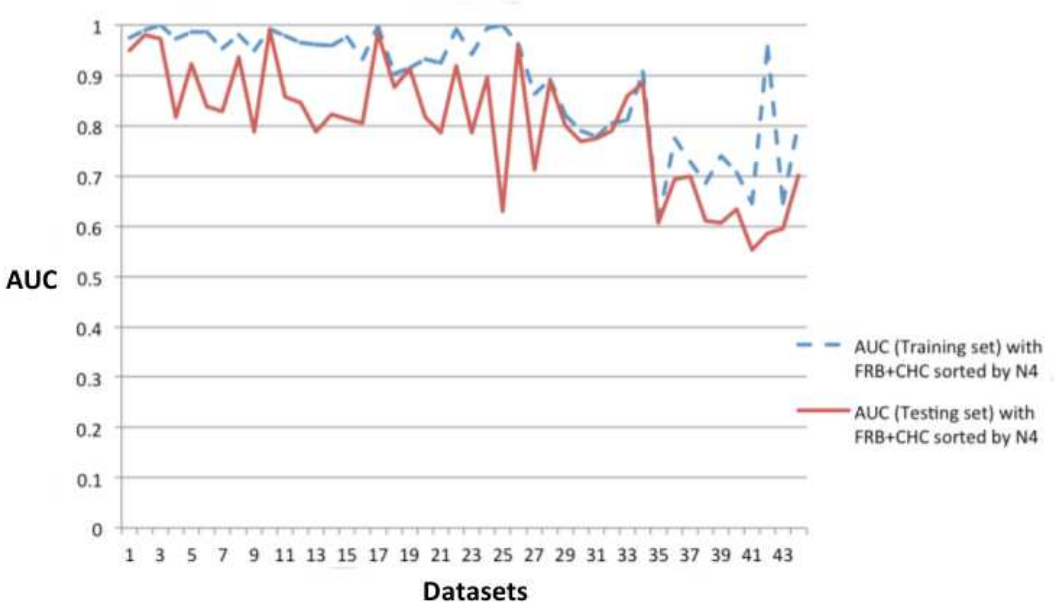
Table 7: The Increase Rate of Number of Support Vectors of the Classification Model formed by SVM.

Dataset	SMOTE	ADASYN	sTL	sSafe	sRST	sCHC	AHC	FRB+CHC
ecoli0347vs56	0.245	0.529	0.418	0.143	0.476	-0.117	0.178	0.026
yeast2vs4	2.662	4.354	2.507	2.981	2.698	1.180	2.377	1.000
ecoli067vs35	0.252	0.506	0.468	0.249	0.465	-0.103	0.215	0.021
ecoli0234vs5	0.272	0.538	0.472	0.122	0.473	-0.097	0.197	0.011
glass015vs2	3.731	3.687	3.193	3.996	3.583	1.705	3.310	1.279
yeast0359vs78	1.384	1.676	1.149	1.434	1.418	0.251	1.361	0.093
yeast0256vs3789	3.555	5.151	3.240	3.669	3.577	1.414	3.416	0.990
yeast02579vs368	2.142	5.240	1.919	2.251	2.188	0.783	1.990	0.521
ecoli046vs5	0.263	0.477	0.460	0.125	0.490	-0.100	0.177	0.031
ecoli0147vs2356	0.229	0.566	0.435	0.170	0.494	-0.103	0.186	0.053
ecoli0267vs35	0.279	0.521	0.490	0.234	0.467	-0.092	0.209	0.040
glass04vs5	1.839	1.477	1.467	4.104	1.242	-0.043	0.122	0.306
ecoli034vs5	0.269	0.526	0.456	0.135	0.477	-0.108	0.163	0.029
ecoli0346vs5	0.286	0.424	0.488	0.140	0.497	-0.094	0.150	0.031
yeast05679vs4	3.500	4.147	3.113	3.561	3.575	1.401	3.474	1.469
vowel0	1.116	1.333	0.752	2.311	0.972	-0.027	0.448	1.631
ecoli067vs5	0.227	0.440	0.391	0.212	0.438	-0.109	0.144	0.055
glass016vs2	3.868	3.927	3.296	4.209	3.660	1.840	3.291	1.530
ecoli0137vs26	0.175	0.404	0.338	0.089	0.439	-0.191	0.160	-0.146
led7digit02456789vs1	3.332	3.322	2.864	4.139	2.838	0.360	2.608	0.563
ecoli0147vs56	0.218	0.487	0.391	0.121	0.522	-0.145	0.147	0.031
glass06vs5	1.494	1.272	1.124	2.924	1.204	0.021	0.077	0.250
glass0146vs2	3.967	4.008	3.396	4.271	3.725	1.915	3.391	1.643
glass2	3.820	3.893	3.221	4.036	3.671	1.881	3.444	1.712
ecoli0146vs5	0.189	0.379	0.361	0.105	0.394	-0.155	0.137	0.014
cleveland0vs4	0.540	0.741	0.712	0.247	0.505	-0.077	0.326	0.008
ecoli01vs5	0.148	0.494	0.309	0.082	0.417	-0.128	0.149	0.032
shuttlec0vsc4	0.221	2.557	0.264	0.001	0.450	-0.109	0.103	1.326
yeast1458vs7	3.121	6.093	2.947	3.233	3.100	1.064	5.311	0.966
glass4	1.968	1.873	0.836	4.223	1.852	0.084	0.675	0.344
ecoli4	2.173	2.998	2.020	2.668	2.394	0.772	1.796	1.012
page_blocks13vs4	0.836	1.042	0.863	0.057	1.198	0.017	0.523	0.110
abalone918	8.586	9.046	8.058	10.025	8.077	3.752	6.750	4.309
glass016vs5	2.084	1.760	1.440	3.498	1.846	0.226	0.492	0.512
shuttlec2vsc4	0.616	1.428	1.388	0.153	1.310	0.282	0.229	0.422
yeast1289vs7	4.846	2.993	4.611	5.170	5.087	1.802	2.825	1.491
glass5	2.307	2.169	1.884	4.623	2.151	0.249	0.524	0.574
yeast2vs8	4.905	8.609	4.920	5.282	5.169	2.101	4.436	1.184
yeast4	3.133	3.547	2.902	3.379	3.173	0.883	2.966	0.919
yeast1vs7	5.196	4.672	4.870	5.523	5.382	2.673	4.088	2.422
yeast5	3.178	3.458	2.617	3.676	3.265	1.578	2.930	2.007
ecoli01vs235	0.286	0.268	0.463	0.153	0.484	-0.096	0.085	0.041
yeast6	6.825	9.553	6.376	7.216	6.900	2.278	6.479	2.632
abalone19	13.156	13.209	12.829	14.221	12.973	5.535	10.986	8.226
Mean	2.351	2.859	2.198	2.708	2.403	**0.776**	1.887	0.948

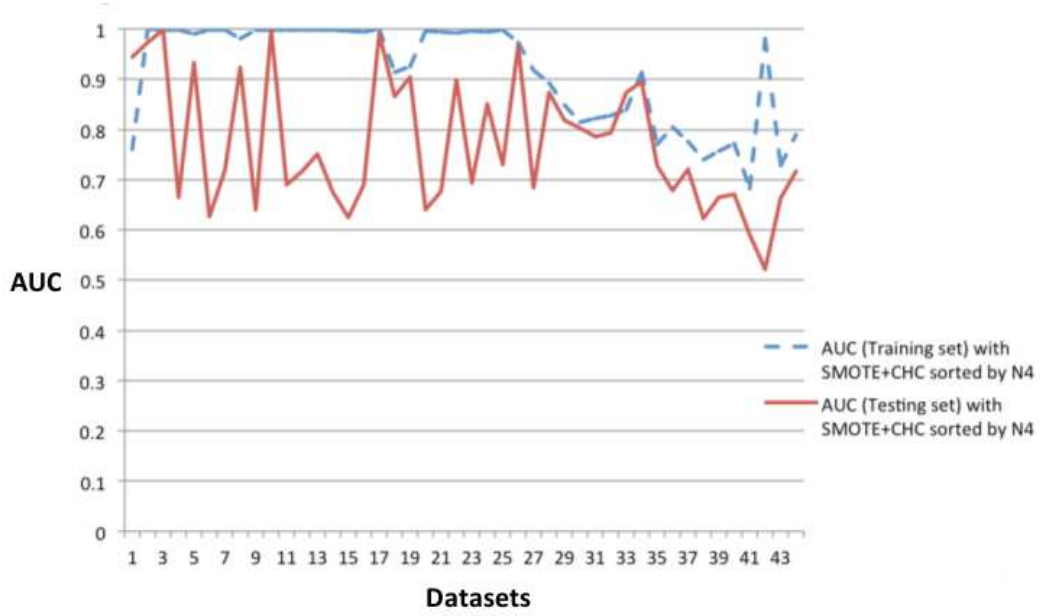
over-fitting problem because the performance of training set and testing set are similar.

Fig. 6 and 7 show an example of the distribution of the positive samples and negative samples after the re-sampling of FRB+CHC and sCHC, respectively. The circle dots correspond to the samples of the majority class. The square dots correspond to the samples of the original minority class. The triangle dots correspond to the synthetic samples. Fig. 7 show that the synthetic samples are generated densely around some of the original minority samples. In contrast, the synthetic samples in Fig. 6 are more evenly distributed in the area of the original minority samples. Therefore, sCHC runs into the over-fitting problem more easily.

Figs. 8 and 9 show the overall results in terms of $F - measure$ and AUC for different classifiers, respectively. Only a small difference of the results for 1NN among all the preprocessing methods is revealed. FRB+CHC obtains the highest value of AUC for both C4.5 and 1NN. An improvement by FRB+CHC in terms of $F - measure$ is shown. In addition, a robust behavior of FRB+CHC is shown when the results of the three classifiers only have a small difference. Most of the preprocessing methods can perform better than the original datasets in terms of the average values of $F - measure$ and AUC. This confirms that preprocessing is an important step to deal with imbalanced datasets.



(a) AUC results with FRB+CHC



(b) AUC results with sCHC

Figure 5: Average AUC results obtained from training and testing sets.

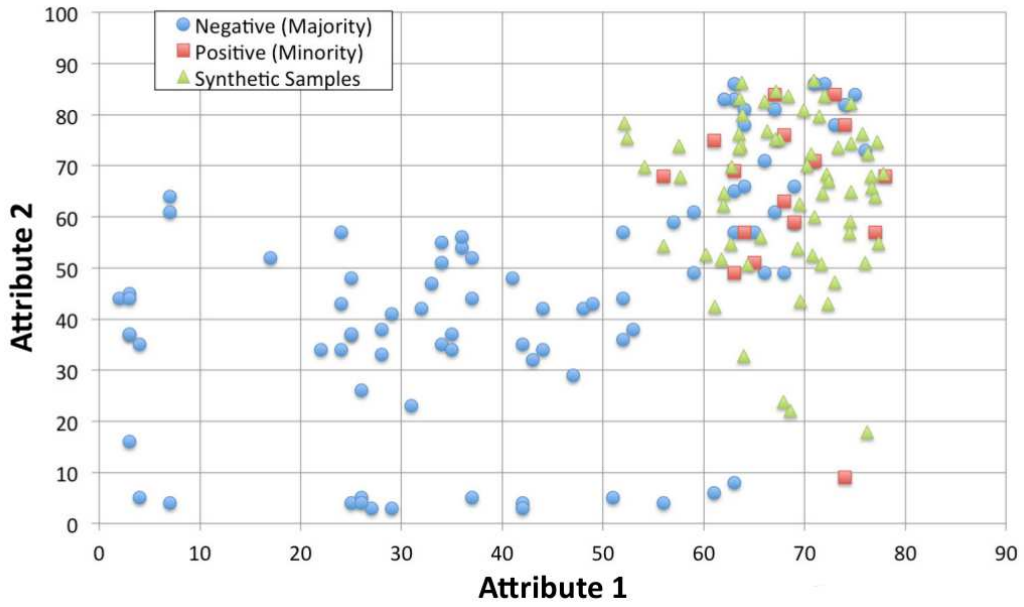


Figure 6: Distribution of the samples after the implementation of FRB+CHC.

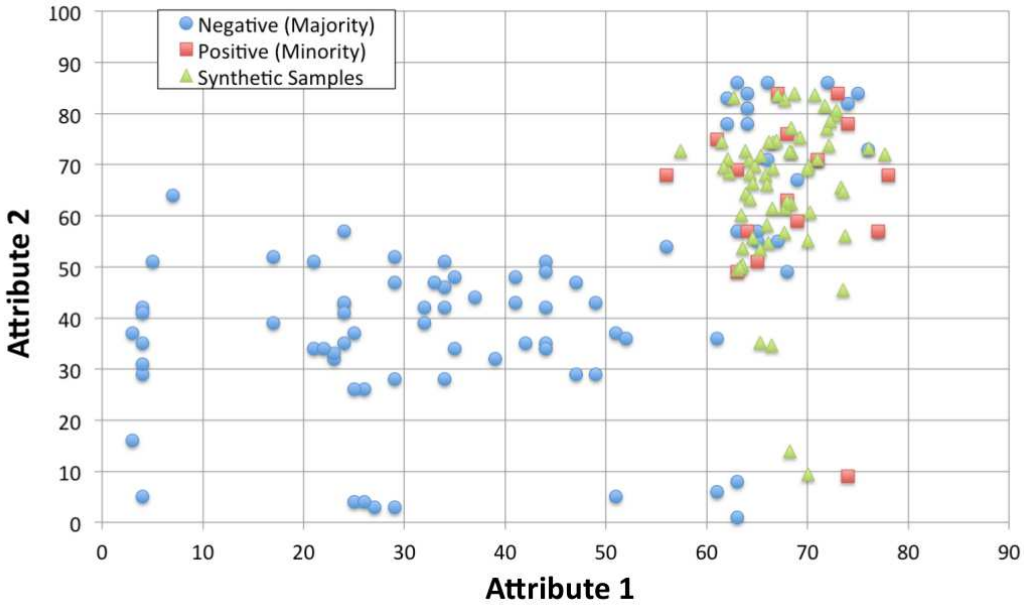


Figure 7: Distribution of the samples after the implementation of sCHC.

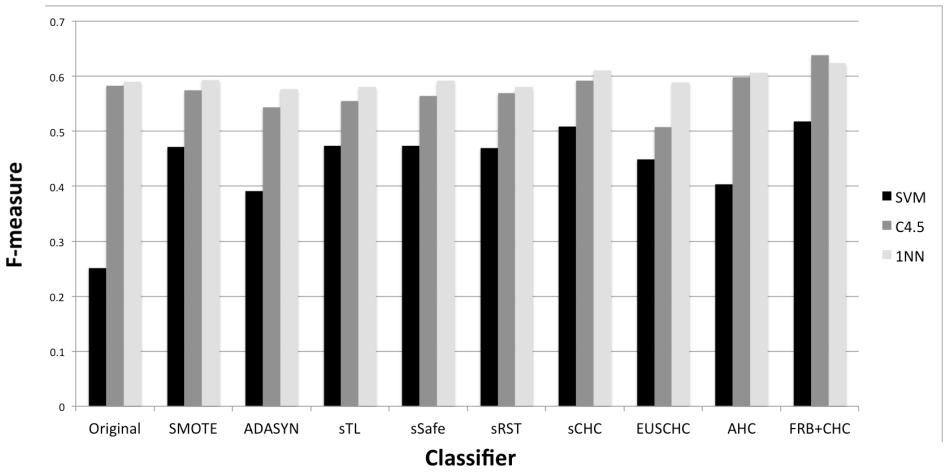


Figure 8: Average $F - measure$ for different classifiers.

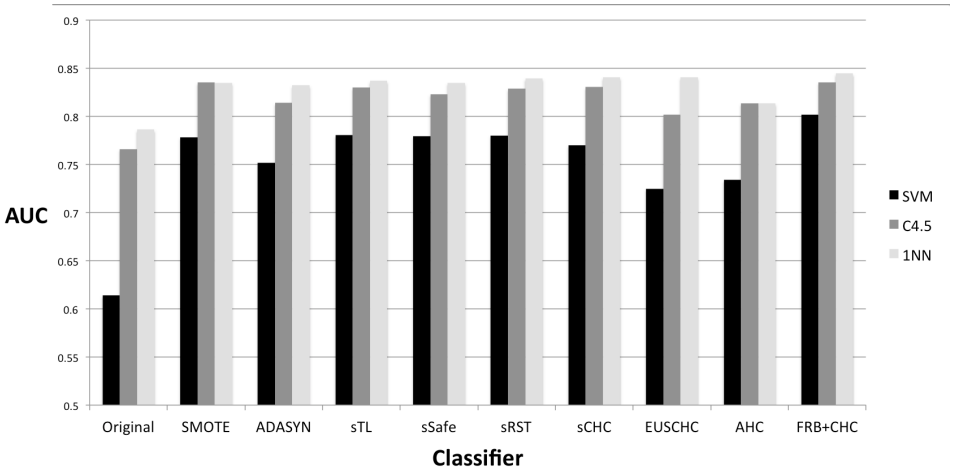


Figure 9: Average AUC for different classifiers.

5. Conclusion

A hybrid re-sampling method developed that is based on both over-sampling and under-sampling has been proposed. The new synthetic samples of the minority class are generated based on fuzzy logic. To minimize the size of datasets, CHC has been employed over the new samples and the majority of samples as a cleaning method to the over-sampled training set.

The proposed sampling method (FRB+CHC) is compared to SMOTE, ADASYN, sTL, sSafe, sRST, sCHC, EUSCHC, and AHC on 44 datasets. To evaluate the performance of these nine sampling methods, the same SVM classifier has been used to obtain the experimental results. It is shown that FRB and FRB+CHC outperforms the other sampling methods on both $F - measure$ and AUC. FRB shows its advantage as an over-sampling method. If data size is not a consideration, then FRB is a better choice of pre-processing method.

FRB+CHC obtains the best ranking by means of AUC. FRB+CHC and sCHC have similar performance in $F - measure$, which indicates that CHC is a good choice of data cleaning method. The AUC results of SMOTE, sTL, sSafe, sRST, and sCHC are similar because they all use SMOTE to perform over-sampling. To show the advantages of the proposed method, the over-sampling rate and the number of support vectors formed from SVM for different methods are also compared. In addition, the C4.5 and 1NN classifiers are used and FRB+CHC shows a

robust behavior among different classifiers. FRB+CHC achieves good results under these criteria, which reflects that FRB+CHC achieves a good balance between accuracy and over-sampling rate. It also has a low impact on the complexity of the learning model. The major reason for this is that CHC only selects the samples to increase the performance of the datasets but does not consider the locations of the samples. Therefore, the most representative samples are selected to form the training sets.

Acknowledgment

The work described in this paper was substantially supported by a grant from The Hong Kong Polytechnic University (Project No. RPKP, RUG7 and G-YN19).

[1] Alcalá-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. (2011). Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2–3):255–287.

[2] Asuncion, A. and Newman, D. (2007). Uci machine learning repository. http://www.ics.uci.edu/ mlearn/MLRepository.html.

[3] Batista, G., Prati, R., and Monard, M. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations*, 6(1):20–29.

[4] Bunkhumpornpat, C., Sinapiromsaran, K., and Lursinsap, C. (2009). Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '09, pages 475–482, Berlin, Heidelberg. Springer-Verlag.

[5] Cano, J., Herrera, F., and Lozano, M. (2003). Using evolutionary algorithms as instance selection for data reduction in kdd: An experimental study. *IEEE Transactions on Evolutionary Computation*, 7(6):561–575.

[6] Cateni, S., Colla, V., and Vannucci, M. (2014). A method for resampling imbalanced datasets in binary classification tasks for real-world problems. *Neurocomputing*, 135:32–41.

[7] Chang, C. and Lin, C. (2011). Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(27):1–27.

[8] Chawla, N., Bowyer, K., Hall, L., and Kegelmeyer, W. (2002). Smote: Synthetic minority over-sampling technique. *Journal Of Artificial Intelligence Research*, 16:321–357.

[9] Chieslak, D., Chawla, N., and Striegel, A. (2006). Combating imbalance in network intrusion datasets. In *Proocedings of the IEEE International Conference on Granular Computing*, pages 732–737.

[10] Cohen, G., Hilario, M., Sax, H., Hugonnet, S., and Geissbuhler, A. (2006). Learning from imbalanced data in surveillance of nosocomial infection. *Artificial Intelligence in Medicine*, 37:7–18.

[11] Eshelman, L. (1991). The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Foundations of Genetic Algorithms*, pages 265–283. Morgan Kaufmann.

[12] Fernández, A., García, S., Jesus, M., and Herrera, F. (2008). A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets. *Fuzzy Sets and Systems*, 159(18):2378–2398.

[13] García, S., Derrac, J., Triguero, I., Carmona, C., and Herrera, F. (2012a). Evolutionary-based selection of generalized instances for imbalanced classification. *Knowledge-Based Systems*, 25(1):3–12.

[14] García, S. and Herrera, F. (2009). Evolutionary undersmpling for classification with imbalanced datasets: proposals and taxonomy. *Evolutionary Computation*, 17(3):275–306.

[15] García, V., Sánchez, J., and Mollineda, R. (2012b). On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowledge-Based Systems*, 25:13–21.

[16] García-Pedrajas, N., Pérez-Rodríguez, J., García-Pedrajas, M., Ortiz-Boyer,

D., and Fyfe, C. (2012). Class imbalance mehtods for translation initiation site recognition in dna sequences. *Knowledge-Based Systems*, 25:22–34.

[17] Gu, Q., Cai, Z., Zhu, L., and Huang, B. (2008). Data mining on imbalanced data sets. In *Proceedings of the International Conference on Advanced Computer Theory and Engineering*, pages 1020–1024.

[18] Guo, H. and Viktor, H. (2004). Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *SIGKDD Explorations*, 6(1):30–39.

[19] Han, H., Wang, W., and Mao, B. (2005). Borderline-smote: A new over-sampling method in imbalanced data sets learning. In *Proceedings of the International Conference on Intelligent Computing (ICIC05)*, pages 878–887. Springer.

[20] Hart, P. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theroy*, 14:515–516.

[21] He, H., Bai, Y., García, E., and Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of International Joint Conference on Neural Networks (IJCNN08)*, pages 1322–1328.

[22] He, H. and García, E. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

[23] Kubat, M. and Matwin, S. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *Proceedings of the 14th International Conference on Machine Learning*, pages 179–186.

[24] Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. In *Proceedings of the 8th Conference on AI in Medicine in Europe: Artificial Intelligence Medicine*, AIME '01, pages 63–66, London, UK, UK. Springer-Verlag.

[25] Lin, Y., Lee, Y., and Wahba, G. (2002). Support vector machines for classification in nonstandard situations. *Machine Learning*, 46(1–3):191–202.

[26] Liu, Y., Chawla, N., Harper, M., Shriberg, E., and Stolcke, A. (2006). A study in machine learning from imbalanced data for sentence boundary detection in speech. *Computer Speech and Language*, 20:468–494.

[27] López, V., Fernández, A., García, S., Palade, V., and Herrera, F. (2013a). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141.

[28] López, V., Fernández, A., Jesus, M., and Herrera, F. (2013b). A hierarchical genetic fuzzy system based on genetic programming for addressing clas-

sification with highly imbalanced and borderline data-sets. *Knowledge-Based Systems*, 38:85–104.

[29] Napierala, K., Stefanowski, J., and Wilk, S. (2010). Learning from imbalanced data in presence of noisy and borderline examples. In *Proceedings of the 7th International Conference on Rough Sets and Current Trends in Computing (RSCT2010)*, pages 158–167.

[30] Peng, L., Qiao, P.-L., and Liu, Y.-C. (2008). A hybrid re-sampling method for svm learning from imbalanced data sets. In *Fuzzy Systems and Knowledge Discovery*, volume 2.

[31] Provost, F. (2000). Machine learning from imbalanced data sets 101. In *Proceedings of the AAAI'2000 Workshop on Imbalanced Data Sets*, pages 1–3.

[32] Provost, F. and Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231.

[33] Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[34] Ramentol, E., Caballero, Y., Bello, R., and Herrera, F. (2012). Smote-rsb*: A hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory. *Knowledge and Information Systems*, 33:245–265.

[35] Raskutti, B. and Kowalczyk, A. (2004). Extreme rebalanceing for svms: a case study. *SIGKDD Explorations*, 6(1):60–69.

[36] Sheskin, D. (2003). *Handbook of parametric and nonparametric statistical procedures*. Chapman & Hall/CRC.

[37] Tomek, I. (1976). Two modifications of cnn. *IEEE Transactions on Systems, Man and Cybernetics*, 6:769–772.

[38] Weiss, G. and Provost, F. (2003). Learning when training data are costly: the effect of class distribution on tree induction. *Journal Of Artificial Intelligence Research*, 19:315–354.

[39] Wilson, D. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics*, 2(3):408–421.

[40] Wong, G., Leung, F., and Ling, S. (2013). A novel evolutionary preprocessing method based on over-sampling and under-sampling for imbalanced datasets. In *Proceedings of the 39th Annual Conference of the IEEE Industrial Electronics Society (IECON 2013)*, pages 2354—2359.

[41] Xu, L., Chow, M., and Taylor, L. (2007). Power distribution fault cause identification with imbalanced data using the data mining-based fuzzy classification e-algorithm. *IEEE Transactions on Power Systems*, 22(1):164–171.

[42] Zhou, L. (2013). Performance of corporate bankruptcy prediction meodels on imbalanced dataset: The effect of sampling methods. *Knowledge-Based Systems*, 41:16–25.