Appl Intell (2012) 36:664–684
DOI 10.1007/s10489-011-0287-y

DBSMOTE: Density-Based Synthetic Minority Over-sampling TEchnique

Chumphol Bunkhumpornpat · Krung Sinapiromsaran · Chidchanok Lursinsap

Published online: 14 April 2011

Abstract A dataset exhibits the class imbalance problem when a target class has a very small number of instances relative to other classes. A trivial classifier typically fails to detect a minority class due to its extremely low incidence rate. In this paper, a new over-sampling technique called DBSMOTE is proposed. Our technique relies on a density-based notion of clusters and is designed to over-sample an arbitrarily shaped cluster discovered by DBSCAN. DBSMOTE generates synthetic instances along a shortest path from each positive instance to a pseudo-centroid of a minority-class cluster. Consequently, these synthetic instances are dense near this centroid and are sparse far from this centroid. Our experimental results show that DBSMOTE improves precision, F-value, and AUC more effectively than SMOTE, Borderline-SMOTE, and Safe-Level-SMOTE for imbalanced datasets.

Keywords Classification · Class imbalance · Over-sampling · Density-based

1 Introduction

Classification [21] is a data mining process that generates a model called a classifier, which describes and distinguishes classes of instances. A derived classifier requires the analysis of a training set, defined as a group of identified instances.

A dataset is considered to be imbalanced if a target class has a very small number of instances compared to other classes. The class imbalance problem [9, 18, 19] has attracted the attention of researchers from various fields. Many applications only consider the two-class case [23, 24, 31]. In this context, the smaller class is called the minority class (the positive class), while the larger class is called the majority class (the negative class). If a dataset has more than two classes, a target class is chosen to be the minority class, and the remaining classes are merged into a single majority class.

Analysts encounter the class imbalance problem in many real-world applications, such as medical decision support system for colon polyp screening [10], retailing bank customer attrition analysis [17], network intrusion detection of rare attack categories [22], automotive engineering diagnosis [26], and vehicle diagnostics [27]. In these domains, a standard classifier needs to accurately detect a minority class. This minority class may have an extremely low rate of incidence, and, accordingly, standard classifiers generally prove inadequate.

One widely used strategy for handling the class imbalance problem involves re-sampling techniques [18, 19], which aim to balance the class distributions of a dataset before feeding the output into a classification algorithm. There are two main re-sampling techniques: over-sampling techniques, which amplify positive instances, and under-sampling techniques, which suppress negative instances. However, over-sampling techniques, which create more specific decision regions, are negatively impacted by the over-fitting problem [30], while under-sampling techniques typically suppress important parts of a dataset. Other strategies

C. Bunkhumpornpat (✉) · K. Sinapiromsaran · C. Lursinsap
Department of Mathematics, Faculty of Science, Chulalongkorn University, Bangkok 10330, Thailand
e-mail: chumphol@chiangmai.ac.th

K. Sinapiromsaran
e-mail: Krung.S@chula.ac.th

C. Lursinsap
e-mail: lchidcha@chula.ac.th

deal with this problem differently, for example a boosting-based algorithm [8] and cost-sensitive learning [14].

Batista, Prati, and Monard (2004) used C4.5 in their experiments [2]. They showed that *AUC* values with various over-sampling techniques are generally superior to those achieved with under-sampling techniques. They concluded that combinations of over-sampling and data cleaning lead to satisfactory *AUC* values when the minority class is small.

This paper is organized as follows. Section 2 briefly overviews related work. Section 3 describes DBSMOTE. Section 4 analyzes DBSMOTE. Section 5 gives an overview of performance metrics and summarizes the experimental results. Section 6 discusses the trade-offs between the various over-sampling techniques. Section 7 summarizes and concludes this work. The Appendix shows all experimental results and their paired t-tests.

2 Related work

2.1 DBSCAN

Ester, Kriegel, Sander, and Xu (1996) designed a density-based clustering algorithm called DBSCAN, Density-Based Spatial Clustering of Applications with Noise [15], which discovers clusters of arbitrary shape. Key definitions in the context of DBSCAN are as follows:

By Definition 1, *dist* returns the distance between instances p and q.

By Definition 2, a core instance relies on the core instance condition but a border instance does not. In addition, an instance is directly density-reachable only from a core instance.

By Definition 3, density-reachable is the transitive extension of directly density-reachable.

By Definition 4, if there is a core instance from which two instances in a cluster are density-reachable, these two instances are density-connected to each other.

Definition 1 (Eps-neighborhood) Let D be a dataset. The Eps-neighborhood of an instance p, denoted by $N_{Eps}(p)$, is defined by $N_{Eps}(p) = \{q \in D \mid dist(p, q) \leq Eps\}$.

Definition 2 (Directly density-reachable) An instance p is directly density-reachable from an instance q w.r.t. *Eps* and *MinPts* if

(1) $p \in N_{Eps}(q)$ and
(2) $|N_{Eps}(q)| \geq MinPts$ (Core instance condition).

Definition 3 (Density-reachable) An instance p is density-reachable from an instance q w.r.t. *Eps* and *MinPts* if there is a chain of instances $p_1, \ldots, p_n, p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

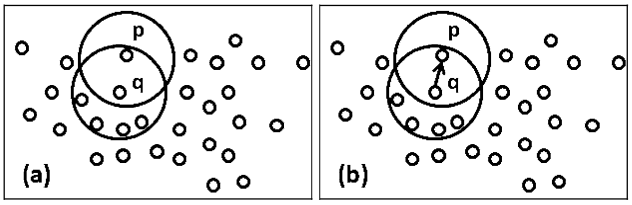

Fig. 1 (**a**) Eps-neighborhood, (**b**) Directly density-reachable

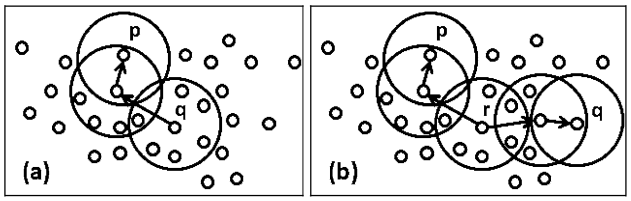

Fig. 2 (**a**) Density-reachable, (**b**) Density-connected

Definition 4 (Density-connected) An instance p is density-connected to an instance q w.r.t. *Eps* and *MinPts* if there is an instance r such that both p and q are density-reachable from r w.r.t. *Eps* and *MinPts*.

In Figs. 1 and 2, a point represents an instance, a circle represents the space obtained by a center point, and an arrow represents directly density-reachable instances. In Fig. 1(a), p is a member of the Eps-neighborhood of q. In Fig. 1(b), p is directly density-reachable from q. In Fig. 2(a), p is density-reachable from q. In Fig. 2(b), p and q are density-connected to each other by r.

According to Definitions 5 and 6, a density-based cluster is a set of instances that are density-connected and that are maximal with respect to density-reachable. This approach assumes that the rest of the instances are noise. In addition, analysts can determine *Eps* and *MinPts* by visualizing and considering a sorted k-dist graph [15].

Definition 5 (Cluster) A cluster C w.r.t. *Eps* and *MinPts* is a non-empty subset of a dataset D satisfying the following conditions:

(1) $\forall p, q$: if $p \in C$ and q is density-reachable from p w.r.t. *Eps* and *MinPts*, then $q \in C$ (Maximality).
(2) $\forall p, q \in C$: p is density-connected to q w.r.t. *Eps* and *MinPts* (Connectivity).

Definition 6 (Noise) Let $C_1, \ldots, C_k$ be the k clusters of a dataset D w.r.t. Eps_i and $MinPts_i$ where $i \in \{1, \ldots, k\}$. The set of instances not belonging to any clusters C_i is considered noise. Noise $= \{p \in D \mid \forall i : p \notin C_i\}$.

2.2 The SMOTE family

The SMOTE family consists of several over-sampling techniques: SMOTE, Borderline-SMOTE, and Safe-Level-

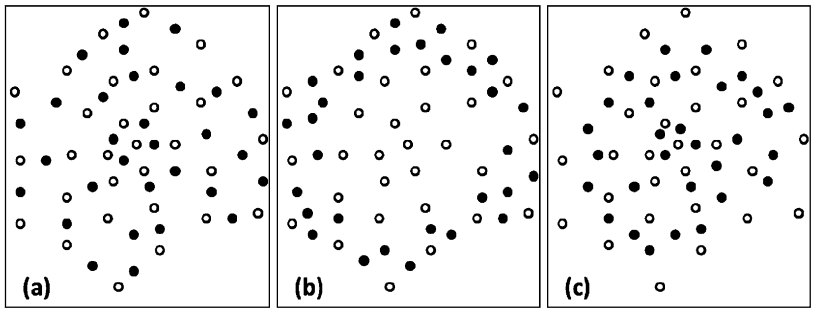

Fig. 3 Over-sampled minority class as processed by (**a**) SMOTE, (**b**) Borderline-SMOTE, and (**c**) Safe-Level-SMOTE

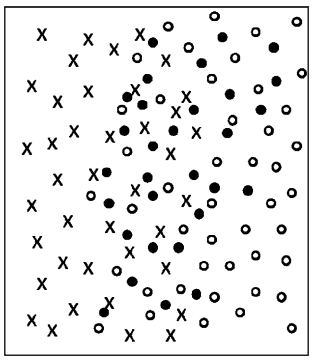

Fig. 4 The over-generalization problem, which impacts SMOTE performance

SMOTE. These techniques upsize a minority class to improve classifier performance at detecting this class.

Chawla, Bowyer, Hall, and Kegelmeyer (2002) designed a state-of-the-art over-sampling technique called SMOTE, Synthetic Minority Over-sampling TEchnique [7], which operates in the feature space rather than in the data space. SMOTE generates synthetic instances along a line segment that join each instance to its selected nearest neighbor. Figure 3(a) illustrates an over-sampled minority class where a white dot represents a positive instance and a black dot represents a synthetic instance. However, SMOTE is negatively impacted by the over-generalization problem because SMOTE blindly generalizes throughout a minority class without considering the majority class, especially in an over-lapping region. Figure 4 illustrates such an over-lapping region, where a cross represents a negative instance. This case is a hybrid between a minority class and a majority class, and it is consequently difficult for a classifier to accurately detect an instance. A superior classifier should treat each region differently.

Han, Wang, and Mao (2005) designed Borderline-SMOTE [16], which operates only on borderline instances in the over-lapping region illustrated in Fig. 3(b)—namely where synthetic instances are most dense. However, the rate of detection of majority classes is disappointing because the classifier mis-detects instances as being positive in this context. A superior classifier should ideally improve the minority class detection rate while not negatively impacting the ability to detect majority classes.

Bunkhumpornpat, Sinapiromsaran, and Lursinsap (2009) designed Safe-Level-SMOTE [6], which generates synthetic instances along the same line segment as SMOTE but locates them closer to a minority class than a majority class. Accordingly, synthetic instances are sparse in an over-lapping region, as illustrated in Fig. 3(c). However, the minority class core is not concentrated, and as a result it may be misclassified. A superior classifier should ideally recognize this core, which is significant.

3 DBSMOTE

In this paper, we propose a new over-sampling technique called DBSMOTE, Density-Based Minority Over-sampling TEchnique, which combines DBSCAN and SMOTE. Our goal is to better address the class imbalance problem.

3.1 Motivation

A real-world dataset that features proximate data clusters is typically defined by a normal distribution, which is dense at the core and sparse towards the borders. Frequently, a classifier correctly detects an unidentified instance within a core because it recognizes this core as a class. Unfortunately a minority class core is too small to be recognized by a classifier, so this core needs to be over-sampled.

The design of DBSMOTE was inspired by various consistent and paradoxical concepts in Borderline-SMOTE. DBSMOTE broadly follows the approach of Borderline-SMOTE, which operates in an over-lapping region, but DBSMOTE precisely over-samples this region to maintain the majority class detection rate. However, DBSMOTE also incorporates a different approach to Borderline-SMOTE, which fails to operate in safe regions. Instead, DBSMOTE over-samples this region to improve minority class detection rates.

3.2 Data structure

In this paper, we define a new data structure called the directly density-reachable graph, which is constructed as an

underlying weighted directed graph [20], as in Definition 7. Note that $\mathbb{R}$ is the set of real numbers.

Definition 7 (Directly density-reachable graph) A directly density-reachable graph of a cluster C discovered by DBSCAN, denoted by $G(C) = (V, E)$, where V is a set of nodes represented as instances in C and E is a set of edges. E is defined as $E = \{(v_1, v_2) \in V \times V |$ an instance v_1 is directly density-reachable from an instance v_2 w.r.t. *Eps* and *MinPts* or vice versa$\}$. Let $w : E \rightarrow \mathbb{R}$ be a weight function where $w(v_1, v_2)$ is equal to distance between nodes $v_1, v_2 \in V$.

A directly density-reachable graph is illustrated in Fig. 5. A black dot represents a core instance, and a white dot represents a border instance. For each edge, at least one node is guaranteed to be a core instance because two border instances cannot be directly density-reachable according to Definition 2.

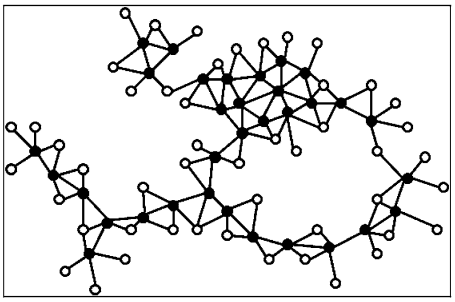

Fig. 5 Directly density-reachable graph

3.3 Algorithm

Figure 6 illustrates the DBSMOTE algorithm, with variables and functions described as follows.

For input and output, p is a positive instance in a cluster i of instances C_i, and s is a synthetic instance in a set i of synthetic instances C'_i.

construct_directly_density-reachable_graph constructs a directly density-reachable graph G from C_i with respect to ε and k.

determine_pseudo-centroid determines a pseudo-centroid c, the nearest instance from a mean of C_i.

Dijkstra runs Dijkstra's algorithm [12] to build a predecessor list π where a given source node is c in G.

retrieve_shortest_path retrieves a shortest path § from p to c by traversing π.

select_random_edge randomly selects an edge e in §.

get_connected_nodes returns a pair of nodes $\{v_1, v_2\}$ connected by e.

generate_random_number generates a random number *gap* from 0 to 1.

In the *for* loop, *atti* is an attribute index, *numattrs* is the number of attributes, and *dif* is the result of subtracting v_2 from v_1 given the same attribute.

The algorithm over-samples along a shortest path from each positive instance to a pseudo-centroid. As a result, the presence of synthetic instances, which are dense near a pseudo-centroid and sparse far from this centroid, will cause the classifier to focus on cluster cores.

The shortest path from a considered instance p to a pseudo-centroid c in a directly density-reachable graph is drawn as a solid line in Fig. 8. Note that not all edges are

Fig. 6 DBSMOTE algorithm

Algorithm: DBSMOTE
Input: a cluster i of positive instances C_i, *Eps* ε, and *MinPts* k
Output: a set i of synthetic instances C'_i

1. $C'_i = \emptyset$
2. $G = construct\_directly\_density\text{-}reachable\_graph(C_i, \varepsilon, k)$
3. $c = determine\_pseudo\text{-}centroid(C_i)$
4. $\pi = Dijkstra(G, c)$
5. $\forall p \in C_i$ {
6. §$= retrieve\_shortest\_path(\pi, p, c)$
7. $e = select\_random\_edge(§)$
8. $(v_1, v_2) = get\_connected\_nodes(e)$
9. for ($atti = 1$ to $numattrs$) {
10. $dif = v_2[atti] - v_1[atti]$
11. $gap = generate\_random\_number(0, 1)$
12. $s[atti] = v_1[atti] + \text{gap} \cdot dif$
13. }
14. $C'_i = C'_i \cup \{s\}$
15. }
16. return C'_i

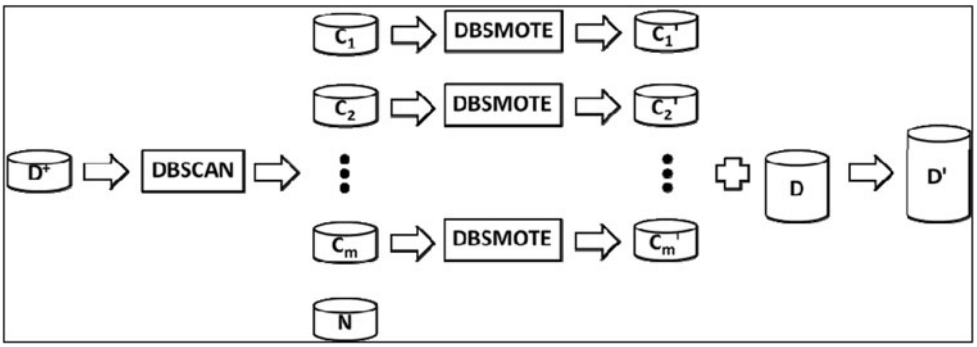

Fig. 7 Over-sampling framework

shown in this figure. Over-sampling along the shortest path avoids the over-lapping problem [28] because this shortest path is the skeleton path [1].

3.4 Framework

A flow diagram for an over-sampling framework integrated with DBSMOTE is illustrated in Fig. 7. DBSCAN begins to produce m disjoint clusters $C_1, C_2, \ldots, C_m$, and detect a set of noise instances N, which erodes in the next step from a minority class D^+. DBSMOTE subsequently generates m sets of synthetic instances: $C'_1, C'_2, \ldots, C'_m$. Eventually, these m sets are merged with an original dataset D to create an over-sampled dataset D'. After this framework terminates, the result is $n - t$ synthetic instances, where n and t are the numbers of instances in D^+ and N, respectively.

4 Analysis

A directly density-reachable graph satisfies the lemmata of similarities of instances. In this graph, the weight of each edge is computed from the distance along the edge. The degree of a node is equal to the number of edges incident to this node. We define two terms: *core node*, which represents a core instance, and *border node*, which represents a border instance.

Lemma 1 is true because Definitions 1 and 2 restrict the distance between two directly density-reachable instances such that it cannot exceed *Eps*.

DBSMOTE over-samples along an edge. The weight parameter reflects the similarity between a synthetic instance and a core instance in a given directly density-reachable graph. In the worst case, the distance between such a pair cannot exceed the similarity-guaranteed distance *Eps*.

Lemma 1 *The maximum weight of an edge in a directly density-reachable graph cannot exceed Eps.*

From the core instance condition in Definition 2, Lemma 2 is true because a core instance obtains at least *MinPts* Eps-neighborhood, and Lemma 3 is true because a border instance does not meet this condition.

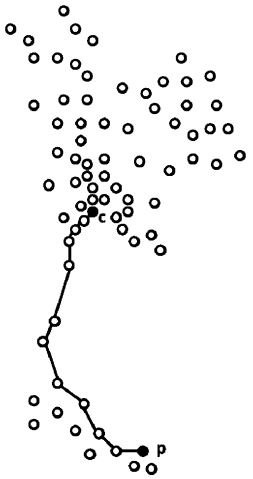

Fig. 8 Shortest path in a directly density-reachable graph

Most visited nodes in a shortest path are core nodes because of their incident edges. Thus, synthetic instances are closer to core instances than border instances. DBSMOTE not only distributes synthetic instances close to a cluster's pseudo-centroid but also locates them close to core instances because these regions contain significant information.

Lemma 2 *The minimum degree of a core node in a directly density-reachable graph cannot be less than MinPts.*

Lemma 3 *The maximum degree of a border node in a directly density-reachable graph must be less than MinPts.*

According to Theorem 1, the running time of DBSMOTE is no lower than that of others in the SMOTE family.

SMOTE is $O(n^2)$ because, for one positive instance, determining the k nearest neighbors and generating synthetic instances have running times $O(n)$ and $O(1)$, respectively.

Borderline-SMOTE is $O(n^2)$ because this technique applies SMOTE, for which the only input is a set of borderline instances.

Safe-Level-SMOTE is $O(n^2)$ because the cost of computing the *safe level* is $O(n)$.

To analyze the time complexity of our over-sampling framework, DBSCAN is $O(n \log n)$ [15], and, consequently,

DBSMOTE is $O(n_i^2)$, which is equivalent to $O(n^2)$, for a cluster i of n_i positive instances. Thus, this framework is $O(n^2)$ because a given dataset will have a finite number of clusters.

Theorem 1 (Time complexity) *DBSMOTE is $O(n^2)$ where n is the number of positive instance in a cluster.*

Proof Let $T(n)$ and $T_i(n)$ be the time complexities of DBSMOTE and for the ith instruction line, respectively.

$T_1(n), T_{10}(n), T_{12}(n), T_{14}(n), T_{16}(n)$ are $O(1)$.

construct_directly_density-reachable_graph is $O(n^2)$, $T_2(n)$, because detecting core instances is $O(n^2)$ and creating edges between these instances and their Eps-neighborhood is $O(n^2)$.

determine_pseudo-centroid is $O(n)$, $T_3(n)$, because computing a cluster mean is $O(n)$ and determining a nearest instance from this mean is $O(n)$.

Dijkstra is $O(n^2)$, $T_4(n)$, because of the time complexity of Dijkstra's algorithm [12].

retrieve_shortest_path is $O(n)$, $T_6(n)$, because, in the worst case, a shortest path visits all nodes in a directly density-reachable graph.

select_random_edge is $O(1)$, $T_7(n)$.

get_connected_nodes is $O(1)$, $T_8(n)$.

generate_random_number is $O(1)$, $T_{11}(n)$.

for at lines 5 and 9 take n and *numattrs* steps, respectively.

$T(n)$ is solved as follows:

$$\begin{aligned} T(n) &= T_1(n) + T_2(n) + T_3(n) + T_4(n) + T_{16}(n) \\ &\quad + n \cdot (T_6(n) + T_7(n) + T_8(n) + T_{14}(n) \\ &\quad + numattrs \cdot (T_{10}(n) + T_{11}(n) + T_{12}(n))) \\ &= O(1) + O(n^2) + O(n) + O(n^2) + O(1) \\ &\quad + n \cdot (O(n) + O(1) + O(1) + O(1) \\ &\quad + numattrs \cdot (O(1) + O(1) + O(1))) \\ &= O(n^2) + n \cdot (O(n) + numattrs) \end{aligned}$$

numattrs is a constant, so we conclude that $T(n) = O(n^2)$. □

According to Theorem 2, the algorithm's correctness is validated because the algorithm cannot terminate unless all of the shortest paths have been completely searched per line 6 of the algorithm. If there is even a single instance that does not have a shortest path from a pseudo-centroid, the output will be incorrect. This theorem confirms that the algorithm is reliable.

Theorem 2 (Correctness) *A directly density-reachable graph of a minority class cluster has a shortest path between each instance and a pseudo-centroid.*

Table 1 Confusion matrix

	Detected positive	Detected negative
Actual positive	*TP*	*FN*
Actual negative	*FP*	*TN*

Proof Begin by assuming the contrary, namely that there exists an instance z for which the shortest path from a minority-class-cluster pseudo-centroid c in a directly density-reachable graph does not exist. Relying on condition 2 in Definition 5, z and c are in the same cluster; thus, z is density-connected to c. By Definition 4, there must be an instance r such that z and c are density-reachable from r. By Definition 3, there is a chain of instances $p_1, \ldots, p_n$, $p_1 = r$, $p_n = z$ such that p_{i+1} is directly density-reachable from p_i, where n is the number of instances in this chain and i is an integer in the range from 1 to $n-1$. By Definition 7, p_{i+1} is directly density-reachable from p_i; thus, an edge connection between p_{i+1} and p_i exists. This sequence is a path between r and z. Because z and c are density-reachable from r, a path between z and c also exists, contradicting our assumption that there is no path between z and c. Thus, there exists a shortest path between each instance and a pseudo-centroid. □

5 Experiments

5.1 Performance measures

Classifier performance metrics are typically evaluated by a confusion matrix, as shown in Table 1. The rows are actual classes, and the columns are detected classes. *TP* (True Positive) is the number of correctly classified positive instances. *FN* (False Negative) is the number of incorrectly classified positive instances. *FP* (False Positive) is the number of incorrectly classified negative instances. *TN* (True Negative) is the number of correctly classified negative instances. The six performance measures; *accuracy*, *precision*, *recall*, *F-value*, *TP rate*, and *FP rate*, are defined by formulae (1) through (6).

$$\text{Accuracy} = (TP + TN)/(TP + FN + FP + TN), \tag{1}$$

$$\text{Recall} = TP/(TP + FN), \tag{2}$$

$$\text{Precision} = TP/(TP + FP), \tag{3}$$

$$\text{F-value} = ((1+\beta)^2 \cdot \text{Recall} \cdot \text{Precision}) / (\beta^2 \cdot \text{Recall} + \text{Precision}), \tag{4}$$

$$TP \text{ Rate} = \text{Sensitivity} = TP/(TP + FN), \tag{5}$$

$$FP \text{ Rate} = 1 - \text{Specificity} = FP/(TN + FP). \tag{6}$$

Table 2 Short descriptions of the experimental UCI datasets

Name	Instance	Attribute	Positive	Negative	% Minority
Pima	768	9	268	500	34.90
Haberman	306	4	81	225	26.47
Glass	214	11	51	163	23.83
Segmentation	2310	20	330	1980	14.29
Satimage	6435	37	626	5809	9.73
Ecoli	336	8	25	311	7.44
Yeast	1484	9	20	1464	1.35

In the domain studied by Lewis and Catlett [25], none of the misclassified positive instances impact *accuracy*, which approaches 100% if all instances are detected as being negative. Accordingly, *accuracy* is not a suitable metric with which to evaluate the class imbalance problem, which aims to achieve superior detection rates in the case of minority classes.

The *F-value* [5] is large when both *recall* and *precision*, which are of relevance to minority classes, are also large. β was set to 1, which means that *recall* is as important as *precision* in our experiments.

ROC, the Receiver Operating Characteristic, summarizes the detection rates over a range of tradeoffs between *TP rate* and *FP rate*. The ROC is a two-dimensional graph in which the x-axis represents the *FP rate* and the y-axis represents the *TP rate*. In addition, *AUC* [4], the Area Under ROC, is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.

5.2 Dataset

We used the seven imbalance datasets from the UCI Repository of Machine Learning Databases [3]: the Pima Indians Diabetes, Haberman's Survival, Glass Identification, Image Segmentation, Landsat Satellite, Ecoli, and Yeast databases. The columns in Table 2 list the dataset name, the number of instances, the number of attributes, the number of positive instances, the number of negative instances, and the minority class percentages, respectively. For each dataset, the target class was chosen as the minority class, and the remaining classes were merged to create a large majority class.

5.3 Validation

In our experiments, we applied 10-fold cross-validation to evaluate *precision*, *recall*, *F-value*, and *AUC* using an original dataset (ORG), SMOTE, Borderline-SMOTE (BORD), Safe-Level-SMOTE (SAFE), and DBSMOTE. We employed the decision tree C4.5 [29], Naïve Bayes [21], support vector machine (SVM) [21], Ripper [11], and k-Nearest Neighbor Classifier (k-NN) approaches [13]. k was set to 5 by default in SMOTE.

Table 3 Average values of all over-sampling percentages from Fig. 9

Technique	Precision	Recall	F-value	AUC
ORG	0.649	0.560	0.601	0.781
SMOTE	0.807	0.925	0.862	0.841
BORD	0.792	0.942	0.861	0.831
SAFE	0.823	**0.958**	**0.885**	0.881
DBSMOTE	**0.856**	0.900	0.877	**0.888**

Table 4 Average values of all over-sampling percentages from Fig. 10

Technique	Precision	Recall	F-value	AUC
ORG	0.453	0.296	0.358	0.609
SMOTE	0.712	0.770	0.739	0.698
BORD	0.727	**0.855**	0.785	0.730
SAFE	0.741	0.830	0.782	0.716
DBSMOTE	**0.800**	0.825	**0.812**	**0.772**

Table 5 Average values of all over-sampling percentages from Fig. 11

Technique	Precision	Recall	F-value	AUC
ORG	0.891	0.804	0.845	0.892
SMOTE	0.952	0.954	0.953	0.943
BORD	0.941	0.954	0.948	0.941
SAFE	0.953	**0.977**	**0.964**	0.956
DBSMOTE	**0.966**	0.957	0.962	**0.958**

We discuss only the experimental results shown in Figs. 9 through 15, in which the x-axis represents the minority class over-sampling percentages and the y-axis represents performance metrics. Averages of all over-sampling percentages are shown in Tables 3 through 9.

Pima has the largest minority class incidence rate of all our experimental datasets. Figure 9 illustrates the experimental results of applying k-NN. DBSMOTE mostly achieves better *precision* and *AUC*, while Safe-Level-SMOTE mostly achieves better *recall* and *F-value*.

The minority class incidence rate is approximately 25% in the Haberman dataset. Figure 10 illustrates the experimental results of applying C4.5. DBSMOTE mostly achieves superior performance except in the case of *recall*, for which Borderline-SMOTE is better.

For the Glass dataset, Non-Window Glass was chosen as the minority class and Window Glass was chosen as the majority class. Figure 11 illustrates the experimental results of applying C4.5. DBSMOTE mostly achieves better *precision*

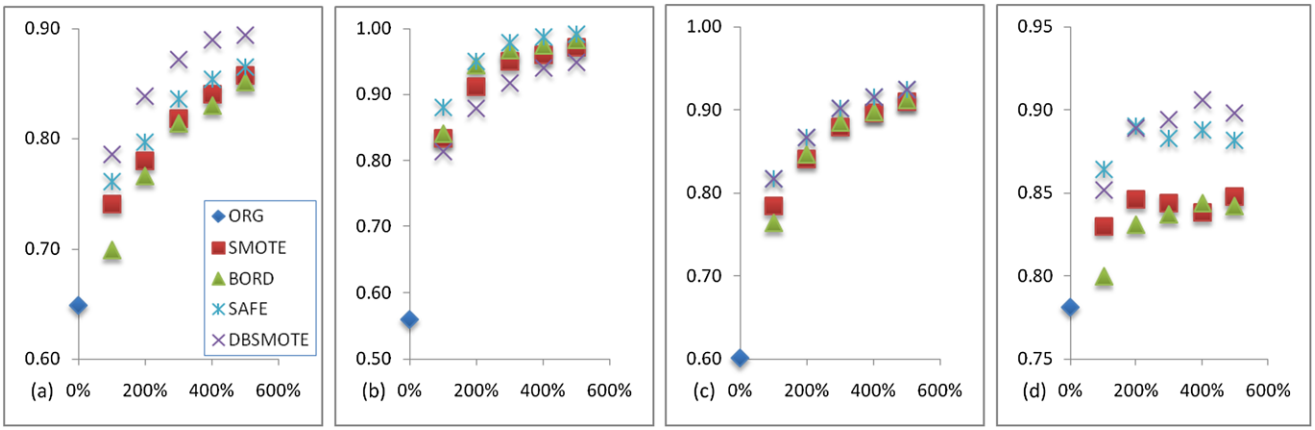

Fig. 9 Experimental results of applying k-NN on Pima: (**a**) Precision, (**b**) Recall, (**c**) F-value, and (**d**) AUC

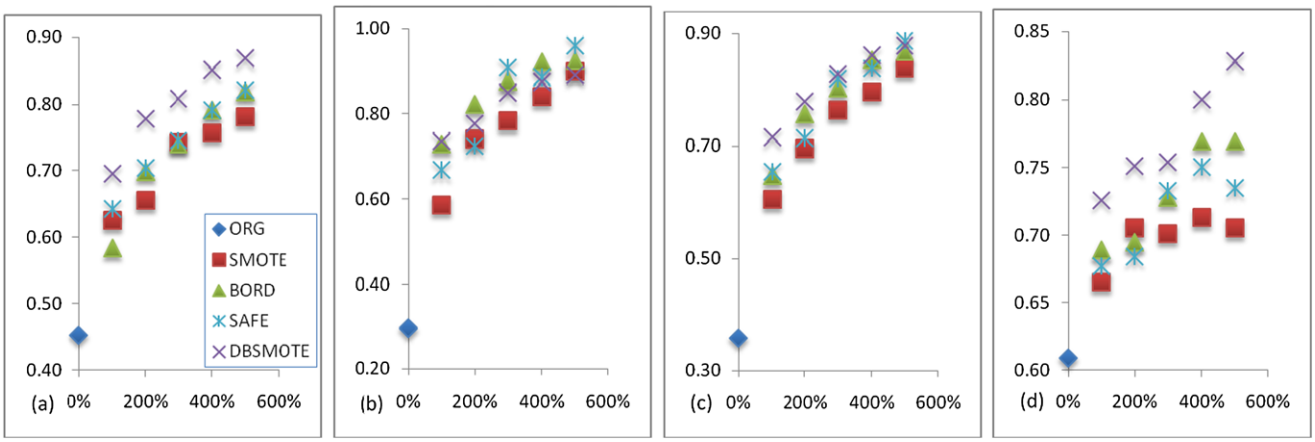

Fig. 10 Experimental results of applying C4.5 on Haberman: (**a**) Precision, (**b**) Recall, (**c**) F-value, and (**d**) AUC

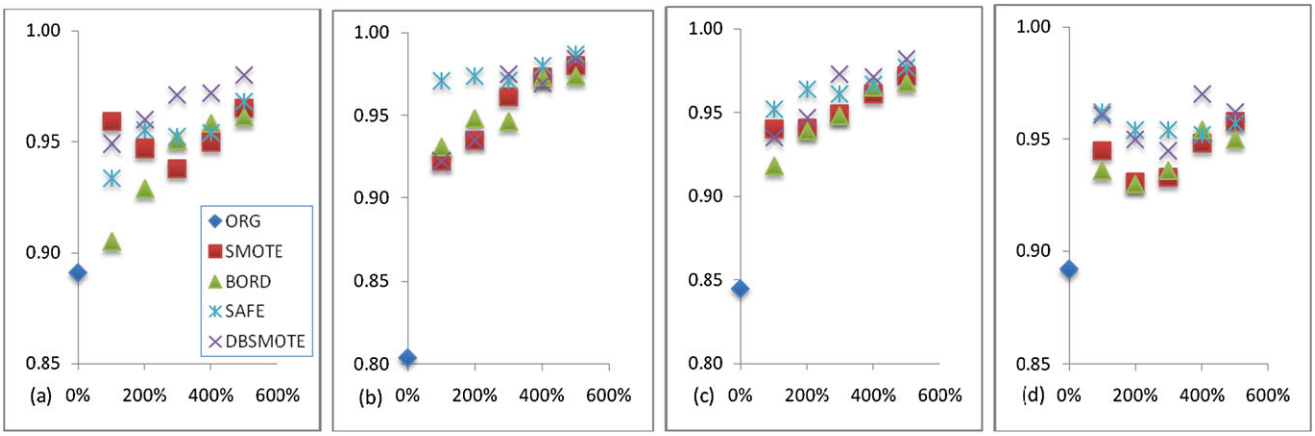

Fig. 11 Experimental results of applying C4.5 on Glass: (**a**) Precision, (**b**) Recall, (**c**) F-value, and (**d**) AUC

and *AUC*, while Safe-Level-SMOTE achieves better *recall* and *F-value*.

In the Segmentation dataset, Window was chosen as the minority class. Figure 12 illustrates the experimental results of applying Naïve Bayes. DBSMOTE mostly achieves superior performance.

In the Satimage dataset, Damp Grey Soil was chosen as the minority class. Figure 13 illustrates the experimental re-

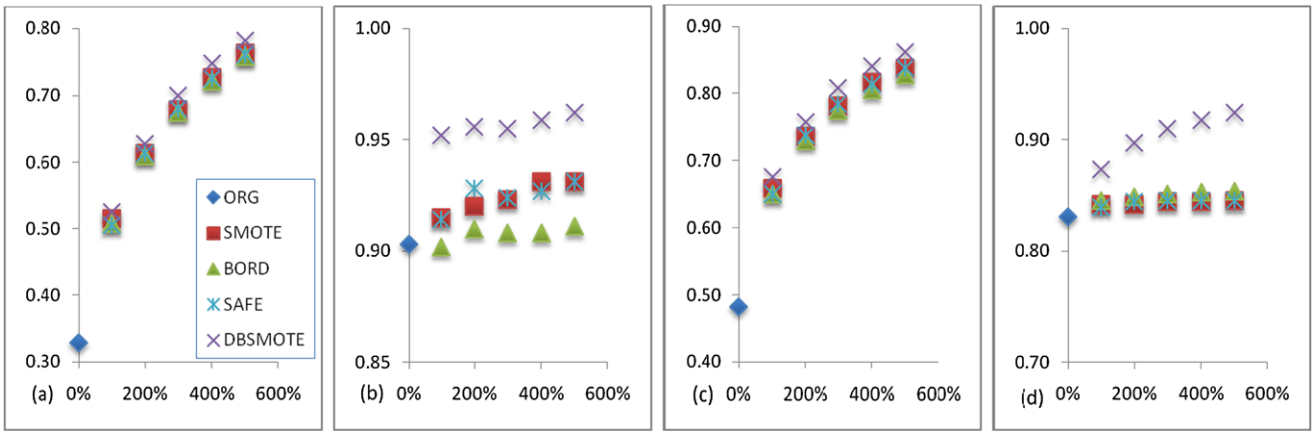

Fig. 12 Experimental results of applying Naïve Bayes on Segmentation: (**a**) Precision, (**b**) Recall, (**c**) F-value, and (**d**) AUC

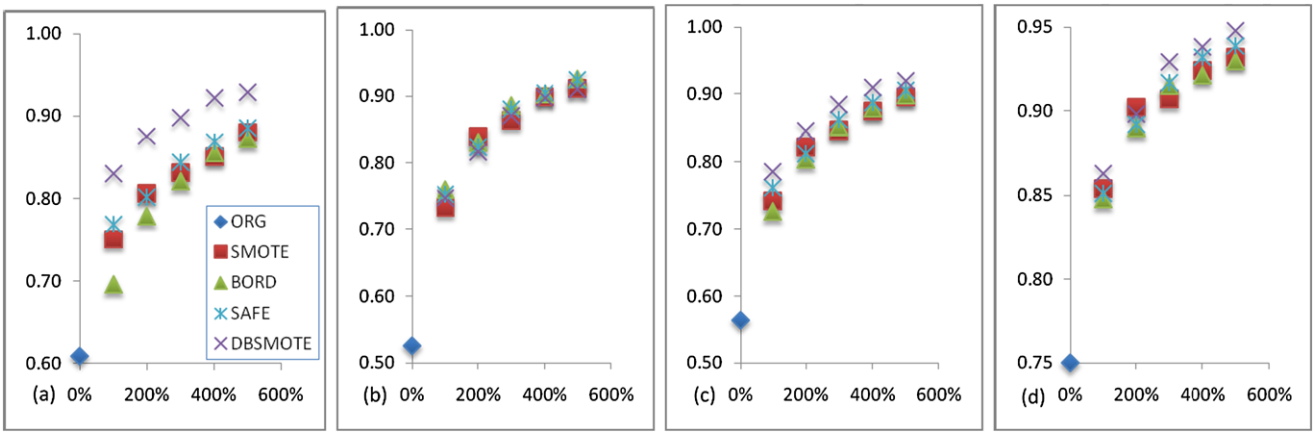

Fig. 13 Experimental results of applying Ripper on Satimage: (**a**) Precision, (**b**) Recall, (**c**) F-value, and (**d**) AUC

Table 6 Average values of all over-sampling percentages from Fig. 12

Technique	Precision	Recall	F-value	AUC
ORG	0.329	0.903	0.482	0.831
SMOTE	0.660	0.924	0.767	0.843
BORD	0.655	0.908	0.757	0.850
SAFE	0.657	0.925	0.765	0.844
DBSMOTE	**0.677**	**0.957**	**0.789**	**0.904**

Table 7 Average values of all over-sampling percentages from Fig. 13

Technique	Precision	Recall	F-value	AUC
ORG	0.609	0.526	0.564	0.750
SMOTE	0.824	0.849	0.836	0.904
BORD	0.805	**0.861**	0.832	0.901
SAFE	0.834	0.857	0.845	0.906
DBSMOTE	**0.891**	0.848	**0.869**	**0.915**

sults of applying Ripper. DBSMOTE mostly achieves superior performance except in the case of *recall*, for which Borderline-SMOTE is better.

In the Ecoli dataset, Outer Membrane was chosen as the minority class. Figure 14 illustrates the experimental results of applying SVM. Safe-Level-SMOTE mostly achieves superior performance.

In the Yeast dataset, Peroxisomal was chosen as the minority class. It was also the most severely imbalanced class among our experimental datasets. Figure 15 illustrates the experimental results of applying SVM. DBSMOTE mostly achieves superior performance.

Paired t-tests were applied to the *F-value* and *AUC* metrics, as illustrated in Tables 10 and 11. For each paired t-test, the null and alternative hypotheses were as follows:

$$H_0 : \mu_1 - \mu_2 = 0,$$

$$H_1 : \mu_1 - \mu_2 \neq 0,$$

where μ_1 is a DBSMOTE mean and μ_2 is a SMOTE, BORD, or SAFE mean. We calculated three paired t-tests:

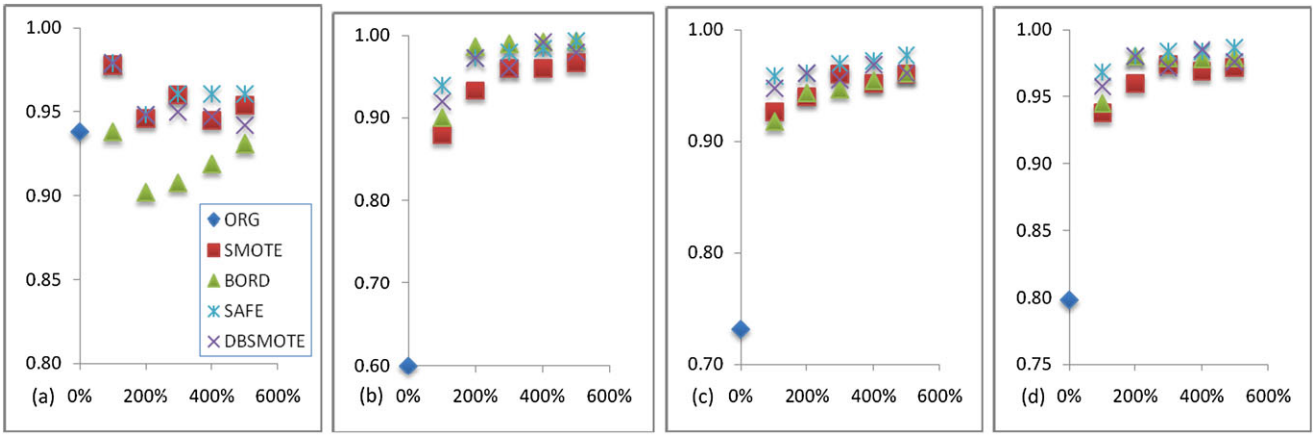

Fig. 14 Experimental results of applying SVM on Ecoli: (**a**) Precision, (**b**) Recall, (**c**) F-value, and (**d**) AUC

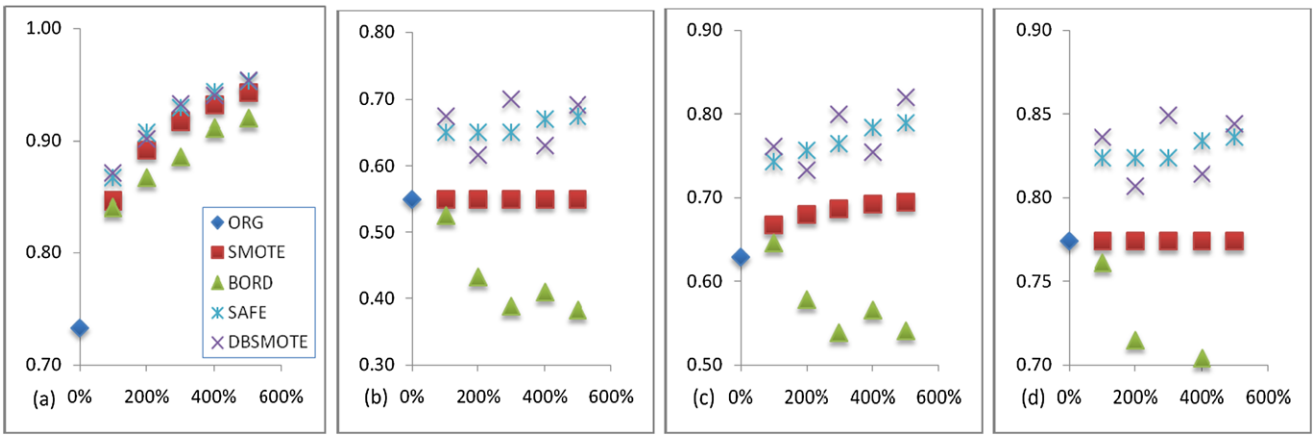

Fig. 15 Experimental results of applying SVM on Yeast: (**a**) Precision, (**b**) Recall, (**c**) F-value, and (**d**) AUC

Table 8 Average values of all over-sampling percentages from Fig. 14

Technique	Precision	Recall	F-value	AUC
ORG	0.938	0.600	0.732	0.798
SMOTE	0.957	0.940	0.948	0.963
BORD	0.920	0.972	0.945	0.972
SAFE	**0.962**	**0.974**	**0.968**	**0.981**
DBSMOTE	0.953	0.965	0.959	0.974

Table 9 Average values of all over-sampling percentages from Fig. 15

Technique	Precision	Recall	F-value	AUC
ORG	0.733	0.550	0.629	0.774
SMOTE	0.906	0.550	0.684	0.774
BORD	0.885	0.428	0.574	0.712
SAFE	**0.920**	0.659	0.768	0.828
DBSMOTE	**0.920**	**0.663**	**0.774**	**0.830**

DBSMOTE to SMOTE, BORD, and SAFE. In these tables, $\Delta\mu$ is a mean difference, $\Delta\sigma^2$ is a variance difference, df is a degree of freedom, t is a t-statistic value, and p is a two-tailed probability value. If $p < \alpha$, H_0 is rejected, which means that there is a difference in means across the paired observations. The significance level α was 0.05.

6 Discussion

There are several interesting questions of relevance to DBSMOTE:

- Which minority class region should be emphasized by an over-sampling technique?

Typically, a dataset is divided into three regions; noise, safe, and over-lapping.

A noise region is located outside a cluster. A classifier often detects noise instances as negative because negative instances encompass these noise instances, and thus an over-sampling technique should not operate in this region.

A safe region is located inside a cluster. A classifier easily recognizes this region because it has sufficient numbers of

Table 10 Paired t-tests on F-value using the indicator variable DBSMOTE

Dataset (classifier)	Method	Variable tested				
		$\Delta\mu$	$\Delta\sigma^2$	df	t	p
Pima	SMOTE	0.0154	−0.000068	4	11.952720	**0.000281**
(k-NN)	Borderline-SMOTE	0.0166	−0.001026	4	3.434564	**0.026425**
	Safe-Level-SMOTE	−0.0078	0.000571	4	−2.750848	0.051330
Haberman	SMOTE	0.0732	−0.003878	4	6.491872	**0.002903**
(C4.5)	Borderline-SMOTE	0.0270	−0.003401	4	2.570846	0.061923
	Safe-Level-SMOTE	0.0302	−0.004538	4	2.165222	0.096324
Glass	SMOTE	0.0090	0.000201	4	1.936492	0.124880
(C4.5)	Borderline-SMOTE	0.0140	−0.000029	4	4.128375	**0.014513**
	Safe-Level-SMOTE	−0.0026	0.000306	4	−0.430590	0.688953
Segmentation	SMOTE	0.0224	0.000449	4	14.281720	**0.000140**
(Naïve Bayes)	Borderline-SMOTE	0.0316	0.000578	4	16.296456	**0.000083**
	Safe-Level-SMOTE	0.0242	0.000020	4	20.166667	**0.000035**
Satimage	SMOTE	0.0328	−0.000556	4	8.114239	**0.001254**
(Ripper)	Borderline-SMOTE	0.0372	−0.001753	4	5.771779	**0.004473**
	Safe-Level-SMOTE	0.0236	−0.000460	4	8.162231	**0.001226**
Ecoli	SMOTE	0.0112	−0.000152	4	2.023362	0.113064
(SVM)	Borderline-SMOTE	0.0142	−0.000207	4	2.826465	**0.047515**
	Safe-Level-SMOTE	−0.0090	0.000004	4	−2.804300	**0.048598**
Yeast	SMOTE	0.0894	0.001136	4	6.384737	**0.003088**
(SVM)	Borderline-SMOTE	0.1996	−0.000634	4	6.416182	**0.003032**
	Safe-Level-SMOTE	0.0058	0.000887	4	0.423189	0.693919

instances. However, for the class imbalance problem, a safe region of a minority class does not contain enough instances, and thus a classifier often misclassifies this region.

An over-lapping region is located around a cluster border, which contains a blend of positive and negative instances. This region is detected with great difficultly because a classifier cannot efficiently distinguish between the two classes, and thus over-sampling in this region might be harmful.

To summarize, for the class imbalance problem, an efficient over-sampling technique should concentrate on a safe region and treat with caution any over-lapping regions.

- How should synthetic instances generated by the SMOTE family be distributed?

SMOTE operates throughout a dataset, and thus synthetic instances are spread throughout every region.

Borderline-SMOTE operates only on a dataset border, and thus synthetic instances are dense in an over-lapping region.

Safe-Level-SMOTE operates throughout a dataset and positions synthetic instances in an over-lapping region close to a safe region. Accordingly, these instances are sparse in an over-lapping region.

DBSMOTE produces more synthetic instances around a dataset core than a dataset border and does not operate within a noise region. Thus these instances are dense in a safe region and are sparse in an over-lapping region.

These distinct distributions cause the effectiveness of these over-sampling techniques to be different.

- How can DBSMOTE achieve a significant *F-value* when Borderline-SMOTE cannot?

For DBSMOTE, a classifier correctly detects a positive instance in a safe region because it has sufficient numbers of detectable synthetic instances. Furthermore, a classifier successfully detects a positive instance in an over-lapping region because most of its synthetic instances tend to be located closer to a minority class than to a majority class. This approach decreases *FP* and *FN*, which in turn increases *precision* and *recall*. Thus, the *F-value* is satisfactory.

For Borderline-SMOTE, a classifier is more likely to misdetect a negative instance located in an over-lapping region as a positive instance because a minority class in this region is dense. This approach not only increases *FP*, which decreases *precision*, but also decreases *FN*, which increases *recall*. Thus a higher *recall* is not enough to improve *F-value* because a lower *precision* will also affect the *F-value*.

Table 11 Paired t-tests on AUC using the indicator variable DBSMOTE

Dataset (classifier)	Method	Variable tested				
		$\Delta\mu$	$\Delta\sigma^2$	df	t	p
Pima	SMOTE	0.0466	0.000386	4	6.285829	**0.003272**
(k-NN)	Borderline-SMOTE	0.0570	0.000118	4	35.349900	**0.000004**
	Safe-Level-SMOTE	0.0064	0.000333	4	1.130312	0.321529
Haberman	SMOTE	0.0740	0.001345	4	5.256297	**0.006270**
(C4.5)	Borderline-SMOTE	0.0418	0.000212	4	6.277380	**0.003288**
	Safe-Level-SMOTE	0.0560	0.000613	4	4.732864	**0.009085**
Glass	SMOTE	0.0146	−0.000024	4	4.673346	**0.009495**
(C4.5)	Borderline-SMOTE	0.0164	−0.000005	4	5.776654	**0.004460**
	Safe-Level-SMOTE	0.0018	0.000085	4	0.387837	0.717891
Segmentation	SMOTE	0.0610	0.000409	4	7.154226	**0.002020**
(Naïve Bayes)	Borderline-SMOTE	0.0540	0.000398	4	7.235460	**0.001936**
	Safe-Level-SMOTE	0.0608	0.000404	4	7.609518	**0.001601**
Satimage	SMOTE	0.0114	0.000278	4	2.604396	0.059771
(Ripper)	Borderline-SMOTE	0.0144	0.000110	4	8.231932	**0.001187**
	Safe-Level-SMOTE	0.0090	−0.000075	4	6.708204	**0.002570**
Ecoli	SMOTE	0.0116	−0.000113	4	2.583525	0.061100
(SVM)	Borderline-SMOTE	0.0020	−0.000127	4	0.559017	0.605967
	Safe-Level-SMOTE	−0.0064	0.000049	4	−2.254300	0.087229
Yeast	SMOTE	0.0560	0.000345	4	6.746498	**0.002516**
(SVM)	Borderline-SMOTE	0.1176	−0.000495	4	7.153465	**0.002021**
	Safe-Level-SMOTE	0.0016	0.000308	4	0.184188	0.862827

7 Conclusion

Many researchers have addressed the class imbalance problem. Over-sampling is a widely used technique because it improves a classifier's ability to detect a low-incidence minority class. Unfortunately, traditional data mining techniques are insufficient.

DBSMOTE executes DBSCAN to discover arbitrarily shaped clusters and then over-samples inside cluster shapes and especially within a pseudo-centroid. Thus, synthetic instances tend to not appear in a majority class. Moreover, these instances are dense near this centroid and sparse far from the centroid.

According to our experimental results, DBSMOTE results in the best *precision*, *F-value*, and *AUC* of any SMOTE family approach when applying the various classifiers. This good performance is due to the synthetic instances of DBSMOTE being generated in the most appropriate places. This fact causes the classifier to concentrate on an important minority class core. The statistical analysis supports our conclusions.

Analysis of DBSMOTE reveals that our technique takes $O(n^2)$, which is no worse than others in the SMOTE family. n is the minority class size. In addition, the DBSMOTE correctness is validated.

Although we provide evidence that DBSMOTE successfully improves detection rates on a minority class, there is considerable room for future work in this line of research. Different clustering algorithms that replace DBSCAN in the framework may improve classifier performance. Pruning a directly density-reachable graph may be of interest. Automatic determination of *Eps*, *MinPts* should be addressed.

Acknowledgements This research is supported by grant funds from the program Strategic Scholarships for Frontier Research Network for the Ph.D. Program Thai Doctoral degree from the Commission on Higher Education, Thailand.

Appendix: Extended experimental results

Figures 16 through 22 show bar graphs to visualize the averages of *F-value* and *AUC*, represented on the y-axis, together with over-sampling percentages on the x-axis. Tables 12 through 18 illustrate the paired t-test results. A * symbol indicates a significant difference ($p < 0.05$) on a paired t-test where the indicator variable is DBSMOTE. The symbols in these figures and tables are defined in Sect. 5.

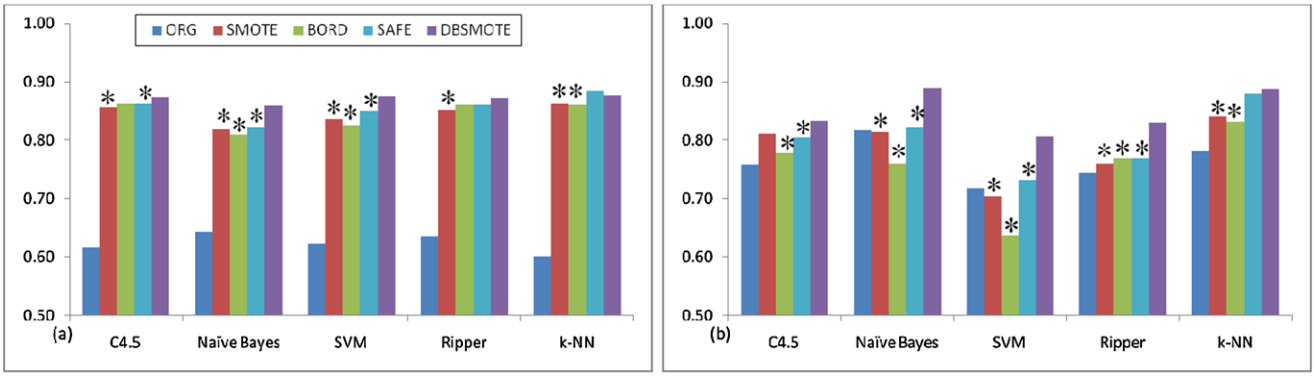

Fig. 16 Experimental results on Pima: (**a**) F-value, (**b**) AUC

Table 12 Paired t-tests on Pima using the indicator variable DBSMOTE

Classifier	Technique	Variable tested				
		$\Delta\mu$	$\Delta\sigma^2$	df	t	p
F-value						
C4.5	SMOTE	0.0168	−0.000046	4	4.548855	**0.010427**
	Borderline-SMOTE	0.0094	−0.000660	4	2.729515	0.052469
	Safe-Level-SMOTE	0.0108	−0.000210	4	3.717513	**0.020519**
Naïve Bayes	SMOTE	0.0408	−0.002277	4	4.236689	**0.013299**
	Borderline-SMOTE	0.0506	−0.003951	4	3.485430	**0.025227**
	Safe-Level-SMOTE	0.0380	−0.000930	4	6.545889	**0.002816**
SVM	SMOTE	0.0380	−0.002107	4	4.459799	**0.011162**
	Borderline-SMOTE	0.0500	−0.001882	4	5.270463	**0.006210**
	Safe-Level-SMOTE	0.0250	−0.001350	4	4.398846	**0.011702**
Ripper	SMOTE	0.0190	−0.001085	4	3.343123	**0.028752**
	Borderline-SMOTE	0.0106	−0.001565	4	1.420804	0.228413
	Safe-Level-SMOTE	0.0112	−0.000926	4	1.939703	0.124420
k-NN	SMOTE	0.0154	−0.000068	4	11.952720	**0.000281**
	Borderline-SMOTE	0.0166	−0.001026	4	3.434564	**0.026425**
	Safe-Level-SMOTE	−0.0078	0.000571	4	−2.750848	0.051330
AUC						
C4.5	SMOTE	0.0230	0.000216	4	2.513999	0.065776
	Borderline-SMOTE	0.0556	0.000158	4	6.217824	**0.003406**
	Safe-Level-SMOTE	0.0292	0.000154	4	5.733210	**0.004584**
Naïve Bayes	SMOTE	0.0750	0.000418	4	8.863490	**0.000895**
	Borderline-SMOTE	0.1286	0.000379	4	10.659090	**0.000439**
	Safe-Level-SMOTE	0.0664	0.000413	4	8.163386	**0.001226**
SVM	SMOTE	0.1024	−0.001479	4	5.663946	**0.004791**
	Borderline-SMOTE	0.1706	−0.006359	4	4.636349	**0.009761**
	Safe-Level-SMOTE	0.0582	−0.000284	4	5.951182	**0.004000**
Ripper	SMOTE	0.0696	−0.000090	4	8.609010	**0.001001**
	Borderline-SMOTE	0.0616	0.000142	4	10.879250	**0.000405**
	Safe-Level-SMOTE	0.0606	−0.000256	4	7.845687	**0.001426**
k-NN	SMOTE	0.0466	0.000386	4	6.285829	**0.003272**
	Borderline-SMOTE	0.0570	0.000118	4	35.349900	**0.000004**
	Safe-Level-SMOTE	0.0064	0.000333	4	1.130312	0.321529

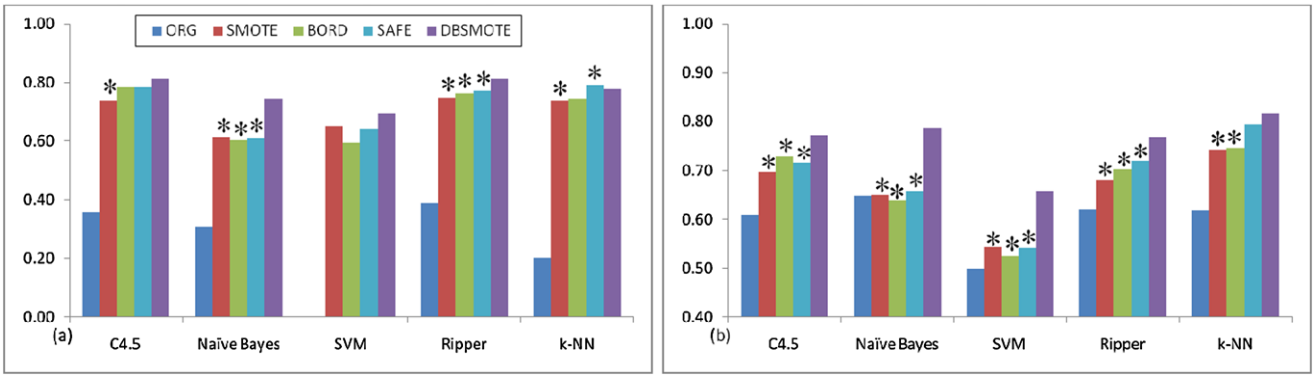

Fig. 17 Experimental results on Haberman: (**a**) F-value, (**b**) AUC

Table 13 Paired t-tests on Haberman using the indicator variable DBSMOTE

Classifier	Technique	Variable tested				
		$\Delta\mu$	$\Delta\sigma^2$	df	t	p
F-value						
C4.5	SMOTE	0.0732	−0.003878	4	6.491872	**0.002903**
	Borderline-SMOTE	0.0270	−0.003400	4	2.570846	0.061923
	Safe-Level-SMOTE	0.0302	−0.004538	4	2.165222	0.096324
Naïve Bayes	SMOTE	0.1296	−0.007579	4	3.837142	**0.018505**
	Borderline-SMOTE	0.1418	−0.025184	4	3.055268	**0.037833**
	Safe-Level-SMOTE	0.1314	−0.010871	4	3.444254	**0.026192**
SVM	SMOTE	0.0402	0.014313	4	1.185774	0.301337
	Borderline-SMOTE	0.0986	−0.014592	4	1.403872	0.233031
	Safe-Level-SMOTE	0.0486	0.010698	4	1.335719	0.252576
Ripper	SMOTE	0.0646	−0.002337	4	8.413056	**0.001093**
	Borderline-SMOTE	0.0484	−0.005066	4	3.324439	**0.029257**
	Safe-Level-SMOTE	0.0400	−0.004840	4	2.819980	**0.047829**
k-NN	SMOTE	0.0400	−0.002592	4	5.557700	**0.005131**
	Borderline-SMOTE	0.0352	−0.006449	4	2.310794	0.081961
	Safe-Level-SMOTE	−0.0146	0.001271	4	−3.921670	**0.017223**
AUC						
C4.5	SMOTE	0.0740	0.001345	4	5.256297	**0.006270**
	Borderline-SMOTE	0.0418	0.000212	4	6.277380	**0.003288**
	Safe-Level-SMOTE	0.0560	0.000613	4	4.732864	**0.009085**
Naïve Bayes	SMOTE	0.1380	0.001065	4	10.064680	**0.000548**
	Borderline-SMOTE	0.1498	0.001006	4	10.726300	**0.000428**
	Safe-Level-SMOTE	0.1298	0.001067	4	10.62152	**0.000445**
SVM	SMOTE	0.1148	−0.000636	4	2.791098	**0.049257**
	Borderline-SMOTE	0.1326	0.001977	4	3.845078	**0.018380**
	Safe-Level-SMOTE	0.1160	−0.000530	4	2.891070	**0.044515**
Ripper	SMOTE	0.0884	0.001208	4	6.025188	**0.003823**
	Borderline-SMOTE	0.0658	−0.000710	4	5.689777	**0.004712**
	Safe-Level-SMOTE	0.0484	0.000839	4	4.979909	**0.007598**
k-NN	SMOTE	0.0772	0.001935	4	5.213836	**0.006455**
	Borderline-SMOTE	0.0714	0.000395	4	13.267800	**0.000187**
	Safe-Level-SMOTE	0.0248	0.001857	4	1.787274	0.148421

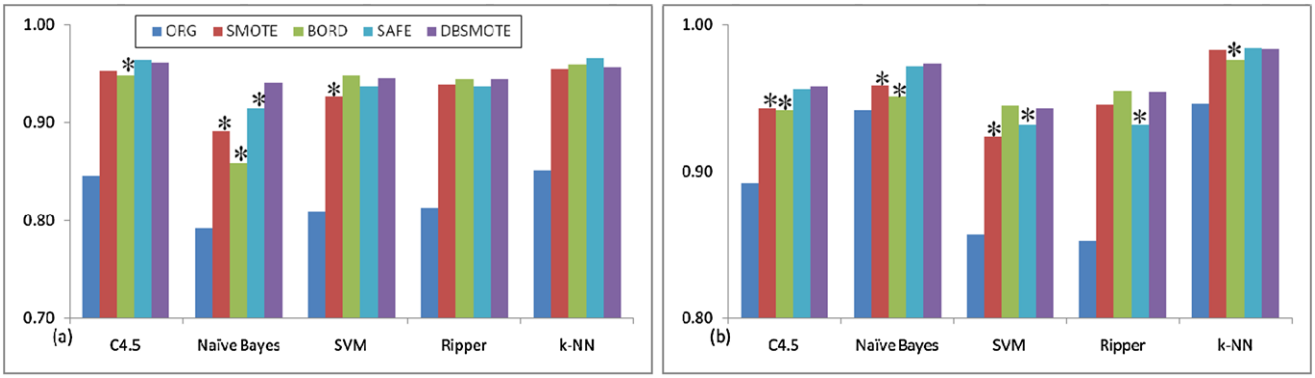

Fig. 18 Experimental results on Glass: (**a**) F-value, (**b**) AUC

Table 14 Paired t-tests on Glass using the indicator variable DBSMOTE

Classifier	Technique	Variable tested				
		$\Delta\mu$	$\Delta\sigma^2$	df	t	p
F-value						
C4.5	SMOTE	0.0090	0.000200	4	1.936492	0.124880
	Borderline-SMOTE	0.0140	−0.000028	4	4.128375	**0.014513**
	Safe-Level-SMOTE	−0.0026	0.000306	4	−0.430590	0.688953
Naïve Bayes	SMOTE	0.0498	−0.000035	4	10.831210	**0.000412**
	Borderline-SMOTE	0.0826	−0.000513	4	13.250390	**0.000187**
	Safe-Level-SMOTE	0.0262	−0.000014	4	12.43397	**0.000241**
SVM	SMOTE	0.0186	−0.000496	4	3.230024	**0.031975**
	Borderline-SMOTE	0.0024	0.000076	4	−1.596460	0.185622
	Safe-Level-SMOTE	0.0088	−0.000530	4	2.022054	0.113233
Ripper	SMOTE	0.0054	−0.000451	4	1.400827	0.233872
	Borderline-SMOTE	0.0002	−0.000218	4	0.097129	0.927296
	Safe-Level-SMOTE	0.0074	0.000117	4	1.296849	0.264432
k-NN	SMOTE	0.0018	−0.000269	4	0.406164	0.705412
	Borderline-SMOTE	−0.0028	0.000237	4	−0.736840	0.502100
	Safe-Level-SMOTE	−0.0090	0.000165	4	−1.982940	0.118403
AUC						
C4.5	SMOTE	0.0146	−0.000024	4	4.673346	**0.009495**
	Borderline-SMOTE	0.0164	−0.000005	4	5.776654	**0.004460**
	Safe-Level-SMOTE	0.0018	0.000085	4	0.387837	0.717891
Naïve Bayes	SMOTE	0.0152	0.000006	4	5.898744	**0.004132**
	Borderline-SMOTE	0.0228	0.000026	4	11.072660	**0.000378**
	Safe-Level-SMOTE	0.0020	0.000017	4	1.450953	0.220415
SVM	SMOTE	0.0190	−0.000184	4	4.023474	**0.015819**
	Borderline-SMOTE	−0.0020	0.000057	4	−1.174440	0.305365
	Safe-Level-SMOTE	0.0114	−0.000120	4	5.338539	**0.005931**
Ripper	SMOTE	0.0092	−0.000257	4	2.102891	0.103316
	Borderline-SMOTE	0.0006	−0.000245	4	−0.124950	0.906594
	Safe-Level-SMOTE	0.0226	−0.000028	4	9.335974	**0.000733**
k-NN	SMOTE	0.0002	0.000011	4	0.179605	0.866194
	Borderline-SMOTE	0.0078	0.000006	4	3.147824	**0.034586**
	Safe-Level-SMOTE	−0.0008	−0.000002	4	−0.267560	0.802268

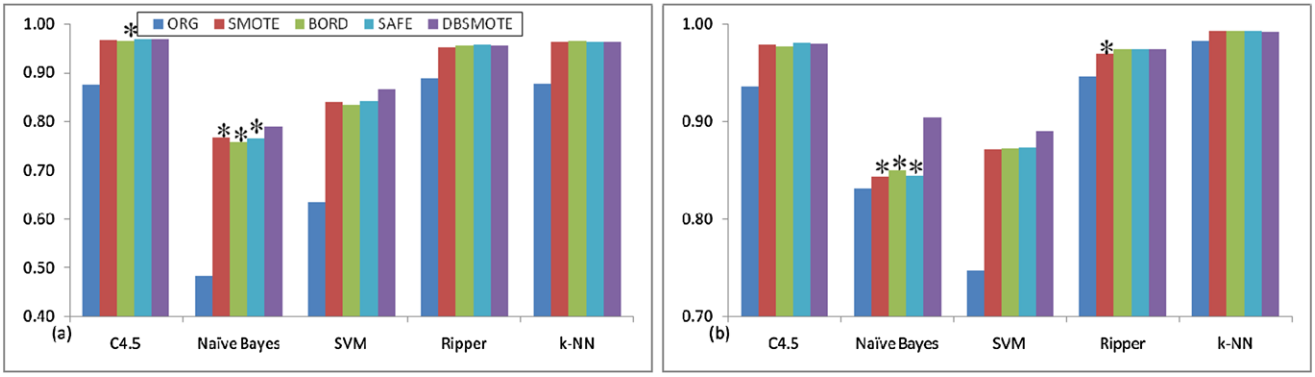

Fig. 19 Experimental results on Segmentation: (**a**) F-value, (**b**) AUC

Table 15 Paired t-tests on Segmentation using the indicator variable DBSMOTE

Classifier	Technique	Variable tested				
		$\Delta\mu$	$\Delta\sigma^2$	df	t	p
F-value						
C4.5	SMOTE	0.0044	−0.000191	4	1.632993	0.177808
	Borderline-SMOTE	0.0046	0.000009	4	5.276562	**0.006185**
	Safe-Level-SMOTE	0.0002	−0.000130	4	0.077615	0.941862
Naïve Bayes	SMOTE	0.0224	0.000449	4	14.281718	**0.000140**
	Borderline-SMOTE	0.0316	0.000578	4	16.296456	**0.000083**
	Safe-Level-SMOTE	0.0242	0.000020	4	20.166667	**0.000035**
SVM	SMOTE	0.0268	−0.004571	4	1.714500	0.161587
	Borderline-SMOTE	0.0324	−0.005935	4	1.624555	0.179582
	Safe-Level-SMOTE	0.0246	−0.004340	4	1.643070	0.175713
Ripper	SMOTE	0.0030	−0.000137	4	1.978141	0.119055
	Borderline-SMOTE	0.0008	0.000064	4	0.644658	0.554258
	Safe-Level-SMOTE	−0.0018	0.000041	4	−1.326980	0.255196
k-NN	SMOTE	0.0004	0.000081	4	0.293294	0.783884
	Borderline-SMOTE	−0.0012	0.000082	4	−0.861550	0.437520
	Safe-Level-SMOTE	0.0000	0.000043	4	0.000000	1.000000
AUC						
C4.5	SMOTE	0.0010	−0.000069	4	0.482243	0.654834
	Borderline-SMOTE	0.0024	0.000003	4	1.530184	0.200715
	Safe-Level-SMOTE	−0.0018	0.000011	4	−1.230450	0.285939
Naïve Bayes	SMOTE	0.0610	0.000409	4	7.154226	**0.002020**
	Borderline-SMOTE	0.0540	0.000398	4	7.235460	**0.001936**
	Safe-Level-SMOTE	0.0608	0.000404	4	7.609518	**0.001601**
SVM	SMOTE	0.0180	−0.001759	4	1.714675	0.161554
	Borderline-SMOTE	0.0178	−0.002218	4	1.287154	0.267473
	Safe-Level-SMOTE	0.0166	−0.001630	4	1.682356	0.167791
Ripper	SMOTE	0.0042	−0.000057	4	3.628247	**0.022194**
	Borderline-SMOTE	0.0004	0.000103	4	0.157378	0.882572
	Safe-Level-SMOTE	0.0000	0.000015	4	0.000000	1.000000
k-NN	SMOTE	−0.0008	0.000003	4	−1.632990	0.177808
	Borderline-SMOTE	−0.0008	0.000006	4	−0.749270	0.495354
	Safe-Level-SMOTE	−0.0014	0.000005	4	−1.870830	0.134702

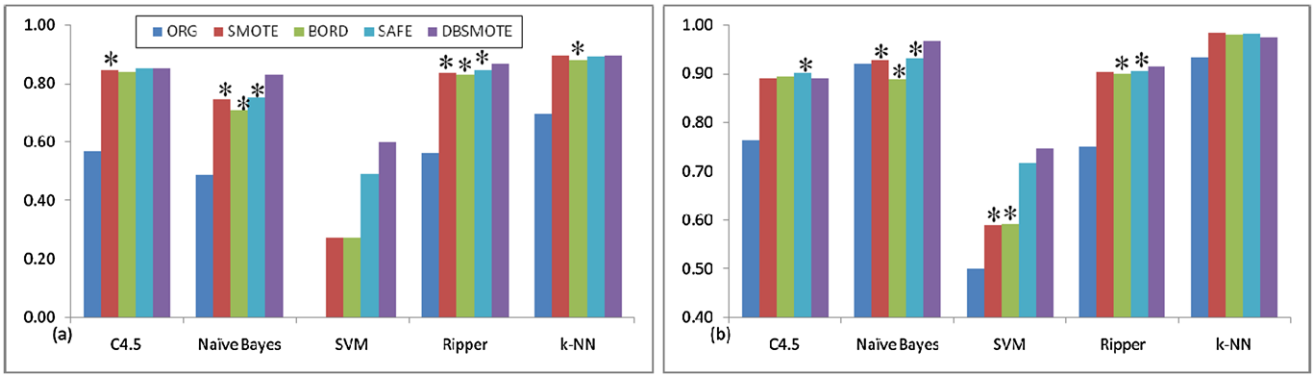

Fig. 20 Experimental results on Satimage: (**a**) F-value, (**b**) AUC

Table 16 Paired t-tests on Satimage using the indicator variable DBSMOTE

Classifier	Technique	Variable tested				
		$\Delta\mu$	$\Delta\sigma^2$	df	t	p
F-value						
C4.5	SMOTE	0.0054	0.000303	4	3.857143	**0.018191**
	Borderline-SMOTE	0.0126	−0.001260	4	2.772078	0.050224
	Safe-Level-SMOTE	−0.0014	−0.000034	4	−0.465120	0.666039
Naïve Bayes	SMOTE	0.0868	0.000805	4	26.006210	**0.000013**
	Borderline-SMOTE	0.1218	0.000742	4	31.832940	**0.000006**
	Safe-Level-SMOTE	0.0812	0.000903	4	21.442800	**0.000028**
SVM	SMOTE	0.3258	−0.012232	4	2.387673	0.075358
	Borderline-SMOTE	0.3260	−0.014095	4	2.367196	0.077056
	Safe-Level-SMOTE	0.1100	−0.097390	4	0.781618	0.478116
Ripper	SMOTE	0.0328	−0.000556	4	8.114239	**0.001254**
	Borderline-SMOTE	0.0372	−0.001753	4	5.771779	**0.004473**
	Safe-Level-SMOTE	0.0236	−0.000460	4	8.162231	**0.001226**
k-NN	SMOTE	−0.0018	0.000414	4	−0.619590	0.569079
	Borderline-SMOTE	0.0150	0.000672	4	3.506434	**0.024752**
	Safe-Level-SMOTE	0.0024	0.000589	4	0.656611	0.547287
AUC						
C4.5	SMOTE	−0.0016	0.000318	4	−0.593820	0.584588
	Borderline-SMOTE	−0.0034	−0.000334	4	−1.077330	0.341970
	Safe-Level-SMOTE	−0.0122	0.000069	4	−3.909130	**0.017407**
Naïve Bayes	SMOTE	0.0770	0.000031	4	11.568810	**0.000319**
	Borderline-SMOTE	0.0384	0.000070	4	11.494740	**0.000327**
	Safe-Level-SMOTE	0.0354	0.000068	4	12.029410	**0.000274**
SVM	SMOTE	0.1574	0.006228	4	3.160120	**0.034180**
	Borderline-SMOTE	0.1552	0.005330	4	3.074298	**0.037137**
	Safe-Level-SMOTE	0.0288	−0.025820	4	0.423166	0.693934
Ripper	SMOTE	0.0114	0.000277	4	2.604396	0.059771
	Borderline-SMOTE	0.0144	0.000110	4	8.231932	**0.001187**
	Safe-Level-SMOTE	0.0090	−0.000075	4	6.708204	**0.002570**
k-NN	SMOTE	−0.0050	0.000062	4	−1.162480	0.309671
	Borderline-SMOTE	−0.0084	0.000064	4	−2.342390	0.079171
	Safe-Level-SMOTE	−0.0078	0.000062	4	−2.220430	0.090569

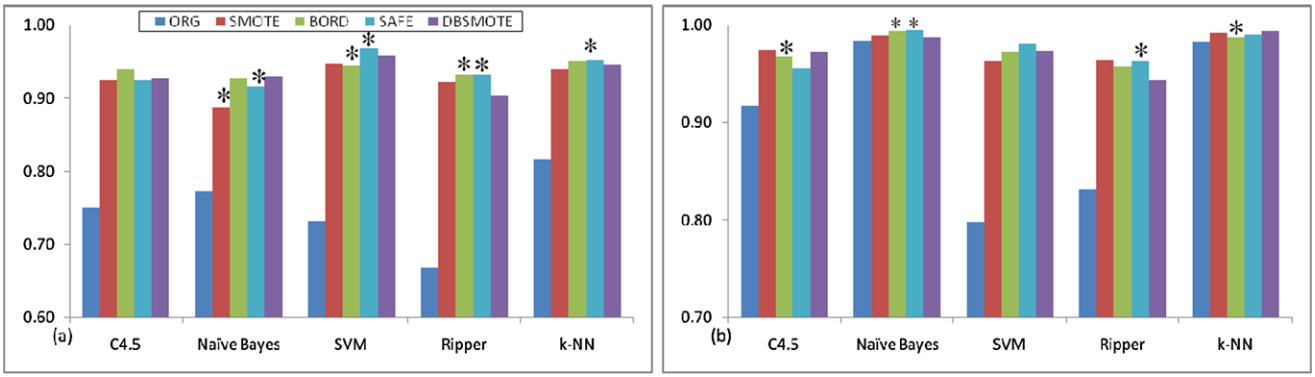

Fig. 21 Experimental results on Ecoli: (**a**) F-value, (**b**) AUC

Table 17 Paired t-tests on Ecoli using the indicator variable DBSMOTE

Classifier	Technique	Variable tested				
		$\Delta\mu$	$\Delta\sigma^2$	df	t	p
F-value						
C4.5	SMOTE	0.0018	0.000604	4	0.243066	0.819910
	Borderline-SMOTE	−0.0132	0.001313	4	−1.466305	0.216450
	Safe-Level-SMOTE	0.0018	−0.002122	4	0.183483	0.863345
Naïve Bayes	SMOTE	0.0436	−0.001155	4	6.355658	**0.003141**
	Borderline-SMOTE	0.0036	0.000334	4	0.366356	0.732654
	Safe-Level-SMOTE	0.0134	0.000400	4	2.790456	**0.049289**
SVM	SMOTE	0.0112	−0.000152	4	2.023362	0.113064
	Borderline-SMOTE	0.0142	−0.000207	4	2.826465	**0.047515**
	Safe-Level-SMOTE	−0.0090	0.000004	4	−2.804300	**0.048598**
Ripper	SMOTE	−0.0188	0.001567	4	−1.840486	0.139520
	Borderline-SMOTE	−0.0286	0.001411	4	−3.373824	**0.027945**
	Safe-Level-SMOTE	−0.0292	0.000436	4	−3.576971	**0.023231**
k-NN	SMOTE	0.0064	−0.000008	4	1.554057	0.195138
	Borderline-SMOTE	−0.0044	0.000104	4	−1.731157	0.158468
	Safe-Level-SMOTE	−0.0072	−0.000135	4	−3.115740	**0.035673**
AUC						
C4.5	SMOTE	−0.0018	0.000016	4	−0.582772	0.591320
	Borderline-SMOTE	0.0056	−0.000003	4	3.055050	**0.037841**
	Safe-Level-SMOTE	0.0172	−0.001113	4	1.326456	0.255353
Naïve Bayes	SMOTE	−0.0018	0.000003	4	−1.087420	0.337989
	Borderline-SMOTE	−0.0068	0.000007	4	−6.106580	**0.003640**
	Safe-Level-SMOTE	−0.0082	0.000010	4	−7.083720	**0.002097**
SVM	SMOTE	0.0116	−0.000113	4	2.583525	0.061100
	Borderline-SMOTE	0.0020	−0.000127	4	0.559017	0.605967
	Safe-Level-SMOTE	−0.0064	0.000049	4	−2.254300	0.087229
Ripper	SMOTE	−0.0208	0.000852	4	−1.800610	0.146134
	Borderline-SMOTE	−0.0142	0.000648	4	−1.854350	0.137296
	Safe-Level-SMOTE	−0.0196	0.000091	4	−3.152280	**0.034438**
k-NN	SMOTE	0.0016	0.000009	4	1.485563	0.211579
	Borderline-SMOTE	0.0060	−0.000030	4	4.140393	**0.014372**
	Safe-Level-SMOTE	0.0032	−0.000451	4	1.777778	0.150072

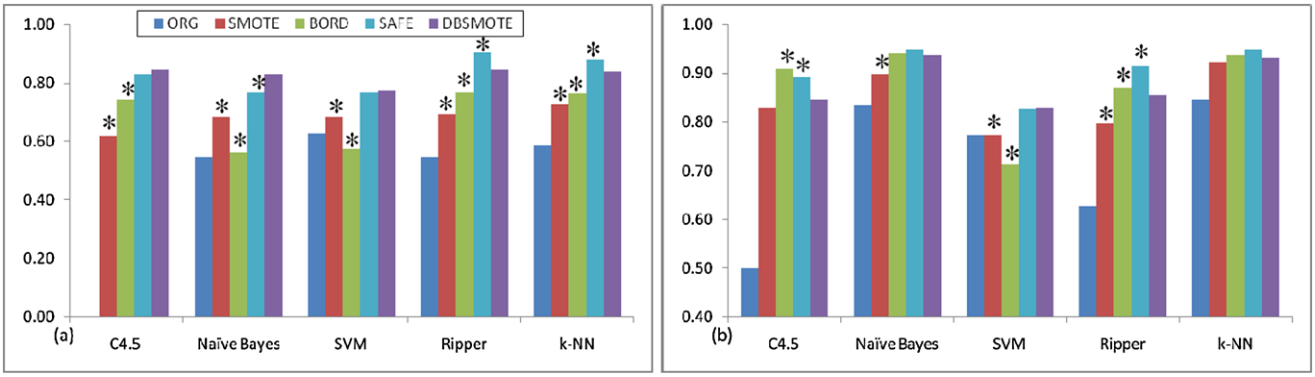

Fig. 22 Experimental results on Yeast (**a**) F-value (**b**) AUC

Table 18 Paired t-tests on Yeast using the indicator variable DBSMOTE

Classifier	Technique	Variable tested				
		$\Delta\mu$	$\Delta\sigma^2$	df	t	p
F-value						
C4.5	SMOTE	0.2266	−0.025624	4	4.321595	**0.012432**
	Borderline-SMOTE	0.1000	−0.003899	4	6.558258	**0.002796**
	Safe-Level-SMOTE	0.0150	−0.007720	4	0.682877	0.532184
Naïve Bayes	SMOTE	0.1462	0.001518	4	10.404710	**0.000482**
	Borderline-SMOTE	0.2666	0.000285	4	7.721343	**0.001515**
	Safe-Level-SMOTE	0.0644	0.001240	4	4.308102	**0.012565**
SVM	SMOTE	0.0894	0.001136	4	6.384737	**0.003088**
	Borderline-SMOTE	0.1996	−0.000634	4	6.416182	**0.003032**
	Safe-Level-SMOTE	0.0058	0.000887	4	0.423189	0.693919
Ripper	SMOTE	0.1524	−0.001344	4	15.437310	**0.000103**
	Borderline-SMOTE	0.0760	−0.000402	4	6.536199	**0.002831**
	Safe-Level-SMOTE	−0.0586	0.000445	4	−8.524140	**0.001039**
k-NN	SMOTE	0.1126	−0.000282	4	7.489047	**0.001700**
	Borderline-SMOTE	0.0752	−0.000603	4	4.867951	**0.008232**
	Safe-Level-SMOTE	−0.0404	−0.001660	4	−2.906850	**0.043816**
AUC						
C4.5	SMOTE	0.0170	−0.004680	4	0.563792	0.603004
	Borderline-SMOTE	−0.0634	0.002637	4	−3.858690	**0.018168**
	Safe-Level-SMOTE	−0.0462	−0.001380	4	−7.083420	**0.002097**
Naïve Bayes	SMOTE	0.0380	−0.000062	4	14.904830	**0.000118**
	Borderline-SMOTE	−0.0040	−0.000155	4	−1.557000	0.194462
	Safe-Level-SMOTE	−0.0096	−0.000400	4	−1.809070	0.144704
SVM	SMOTE	0.0560	0.000344	4	6.746498	**0.002516**
	Borderline-SMOTE	0.1176	−0.000495	4	7.153465	**0.002021**
	Safe-Level-SMOTE	0.0016	0.000308	4	0.184188	0.862827
Ripper	SMOTE	0.0586	−0.000375	4	9.328656	**0.000735**
	Borderline-SMOTE	−0.0146	−0.000424	4	−2.964200	**0.041382**
	Safe-Level-SMOTE	−0.0600	0.001026	4	−9.393360	**0.000716**
k-NN	SMOTE	0.0084	−0.000283	4	0.892104	0.422754
	Borderline-SMOTE	−0.0062	−0.000594	4	−0.781500	0.478178
	Safe-Level-SMOTE	−0.0172	0.000804	4	−1.356220	0.246532

References

1. Bai X, Yang X, Yu D, Latecki LJ (2008) Skeleton-based shape classification using path similarity. Int J Pattern Recognit Artif Intell 22(4):733–746
2. Batista GEAPA, Prati RC, Monard MC (2004) A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor 6(1):20–29
3. Blake CL, Merz CJ (2009) UCI Repository of machine learning databases. http://archive.ics.uci.edu/ml/. Department of Information and Computer Sciences, University of California, Irvine, California, USA
4. Bradley AP (1997) The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognit 30(6):1145–1159
5. Buckland M, Gey F (1994) The relationship between recall and precision. J Am Soc Inf Sci 45(1):12–19
6. Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C (2009) Safe-level-SMOTE: safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In: Theeramunkong T, Kijsirikul B, Cercone N, Ho T-B (eds) 13th Pacific-Asia conference on knowledge discovery and data mining, Bangkok, Thailand. Lecture notes in artificial intelligence, vol 5476. Springer, Heidelberg, pp 475–482
7. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:341–378
8. Chawla NV, Lazarevic A, Hall LO, Bowyer KW (2003) SMOTEBoost: improving prediction of the minority class in boosting. In: The 7th European conference on principles and practice of knowledge discovery in databases, Cavtat-Dubrovnik, Croatia, pp 107–119
9. Chawla NV, Japkowicz N, Kolcz A (2004) SIGKDD Explor 6(1):1–6. Editorial: Special Issue on Learning from imbalanced data sets
10. Chiang I-J, Shieh M-J, Hsu JY, Wong J-M (2005) Building a medical decision support system for colon polyp screening by using fuzzy classification trees. Appl Intell 22(1):61–75. Special Issue: Foundations and Advances in Data Mining
11. Cohen WW (1995) Fast effective rule induction. In: 12th international conference on machine learning, Lake Tahoe, California, USA, pp 115–123
12. Corman TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to algorithms, 2nd edn. MIT Press, Cambridge
13. Cover T, Hart PE (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13(1):21–27
14. Domingos P (1999) Metacost: a general method for making classifiers cost-sensitive. In: The 5th ACM SIGKDD international conference on knowledge discovery and data mining, San Diego, California, USA, pp 155–164
15. Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: The 2nd international conference on knowledge discovery and data mining, Portland, Oregon, USA, pp 226–231
16. Han H, Wang W-Y, Mao B-H (2005) Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Huang D-S, Zhang X-P, Huang G-B (eds) The 2005 international conference on intelligent computing, Hefei, China. Lecture notes in computer science, vol 3644. Springer, Heidelberg, pp 878–887
17. Hu X (2005) A data mining approach for retailing bank customer attrition analysis. Appl Intell 22(1):47–60. Special Issue: Foundations and Advances in Data Mining
18. Japkowicz N (2000) The class imbalance problem: significance and strategies. In: 2000 international conference on artificial intelligence, Las Vegas, Nevada, USA, pp 111–117
19. Japkowicz N (2003) Class imbalance: are we focusing on the right issue? In: 20th international conference on machine learning, Washington, District of Columbia, USA, pp 17–23
20. Jungnickel D (2003) Graphs, networks and algorithms. Springer, Heidelberg
21. Kamber M, Han J (2000) Data mining: concepts and techniques, 2nd edn. Morgan Kaufman, San Mateo
22. Khor K-C, Ting C-Y, Phon-Amnuaisuk S (2010) A cascaded classifier approach for improving detection rates on rare attack categories in network intrusion detection. Appl Intell. doi:10.1007/s10489-010-0263-y
23. Kubat M, Matwin S (1997) Addressing the curse of imbalanced training sets: one-sided selection. In: 14th international conference on machine learning, Nashville, Tennessee, USA, pp 179–186
24. Kubat M, Holte R, Matwin S (1997) Learning when negative examples abound. In: 9th European conference on machine learning, Prague, Czech Republic, pp 146–153
25. Lewis DD, Catlett J (1994) Heterogeneous uncertainty sampling for supervised learning. In: 11th international conference on machine learning, New Brunswick, New Jersey, USA, pp 148–156
26. Lu Y, Chen TQ, Hamilton B (1998) A fuzzy diagnostic model and its application in automotive engineering diagnosis. Appl Intell 9(3):231–243
27. Murphey YL, Chen ZH, Feldkamp LA (2008) An incremental neural learning framework and its application to vehicle diagnostics. Appl Intell 28(1):29–49
28. Prati RC, Batista GEAPA, Monard MC (2004) Class imbalances versus class overlapping: an analysis of a learning system behavior. In: Monroy R, Arroyo G, Sucar LE, Sossa H (eds) 3rd Mexican international conference on artificial intelligence, Mexico City, Mexico. Lecture notes in artificial intelligence, vol 2972, pp 312–321
29. Quinlan JR (1992) C4.5: programs for machine learning. Morgan Kaufmann, San Mateo
30. Tetko IV, Livingstone DJ, Luik AI (1995) Neural network studies. 1. Comparison of overfitting and overtraining. J Chem Inf Comput Sci 35(5):826–833
31. Tomek I (1976) Two modifications of CNN. IEEE Trans Syst Man Cybern 6(11):769–772

Chumphol Bunkhumpornpat received his B.Eng. in Computer Engineering from Rangsit University and his M.S. in Computer Science from Chiang Mai University. He was a lecturer in the Department of Computer Science, Chiang Mai University. He is currently a Ph.D. candidate in Computer Science at Chulalongkorn University. His research focus is Data Mining-especially in the Class Imbalance Problem and Clustering.

Krung Sinapiromsaran received his B.S. in Mathematics from Chulalongkorn University, his M.S. and Ph.D. in Computer Science from the University of Wisconsin-Madison. He is currently an Assistant Professor in the Department of Mathematics, Chulalongkorn University. His ongoing research works are related to theory and application of Mathematical Programming, Knowledge Discovery and Discrete Algorithms.

Chidchanok Lursinsap received his B.Eng. in Computer Engineering from Chulalongkorn University, his M.S. and Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign. He was a Lecturer at the Department of Computer Engineering, Chulalongkorn University from 1978 to 1979, and a visiting Assistant Professor from 1986 to 1987 with the Department of Computer Science and Center for Advanced Studies, University of Illinois at Urbana-Champaign. Presently, he is a Professor at the department of Mathematics, Chulalongkorn University. His research interests include Design Automation, Silicon Compilation, Neural Networks, Computer Architecture, and Artificial Intelligence.