Arabian Journal for Science and Engineering
https://doi.org/10.1007/s13369-019-04336-1

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

HCAB-SMOTE: A Hybrid Clustered Affinitive Borderline SMOTE Approach for Imbalanced Data Binary Classification

Hisham Al Majzoub[1] · Islam Elgedawy[2] · Öykü Akaydın[3] · Mehtap Köse Ulukök[4]

Received: 12 September 2019 / Accepted: 31 December 2019

Abstract
Binary datasets are considered imbalanced when one of their two classes has less than 40% of the total number of the data instances (i.e., minority class). Existing classification algorithms are biased when applied on imbalanced binary datasets, as they misclassify instances of minority class. Many techniques are proposed to minimize the bias and to increase the classification accuracy. Synthetic Minority Oversampling Technique (SMOTE) is a well-known approach proposed to address this problem. It generates new synthetic data instances to balance the dataset. Unfortunately, it generates these instances randomly, leading to the generation of useless new instances, which is time and memory consuming. Different SMOTE derivatives were proposed to overcome this problem (such as Borderline SMOTE), yet the number of generated instances slightly changed. To overcome such problem, this paper proposes a novel approach for generating synthesized data instances known as Hybrid Clustered Affinitive Borderline SMOTE (HCAB-SMOTE). It managed to minimize the number of generated instances while increasing the classification accuracy. It combines undersampling for removing majority noise instances and oversampling approaches to enhance the density of the borderline. It uses *k*-means clustering on the borderline area and identify which clusters to oversample to achieve better results. Experimental results show that HCAB-SMOTE outperformed SMOTE, Borderline SMOTE, AB-SMOTE and CAB-SMOTE approaches which were developed before reaching HCAB-SMOTE, as it provided the highest classification accuracy with the least number of generated instances.

Keywords Imbalanced data · Borderline SMOTE · Oversampling · SMOTE · AB-SMOTE · *k*-means clustering

1 Introduction

Binary supervised learning is the use of datasets that contain instances belonging into two different classes in data mining programs for classification purposes. These datasets suffer from a data level problem, called imbalanced data. This happens when one of the classes has more instances than the other, dividing the classes into majority and minority classes.

The majority class, called the negative class, is the class that has the largest number of instances (e.g., more than 60% of instances within the dataset), whereas the minority class is called the positive class and it represents the class that has the least number of instances (e.g., less than 40% of the instances within the dataset). Imbalanced datasets lead to a bias while building the classification model toward the larger class, where the model would classify most of the new instances in the majority class. The imbalanced binary class problem is found in different fields such as categorization [1], medical field [2], customer churn

✉ Hisham Al Majzoub
hisham.m@hotmail.it

Islam Elgedawy
elgedawy@metu.edu.tr

Öykü Akaydın
oakaydin@ciu.edu.tr

Mehtap Köse Ulukök
mehtap.kose@adakent.edu.tr

1 Management Information Systems Department, School of Applied Sciences, Cyprus International University, Via Mersin 10, Nicosia, Turkey

2 Computer Engineering Department, Middle East Technical University, Northern Cyprus Campus, 99738 Kalkanlı, Guzelyurt, Mersin 10, Turkey

3 Department of Computer Engineering, Cyprus International University, Via Mersin 10, Nicosia, Turkey

4 Department of Software Engineering, University of City Island, Via Mersin 10, Famagusta, Turkey

Published online: 10 January 2020

prediction [3], wine quality [4] and others. The intended class to be classified is usually the minority class. Therefore, this research aims to increase the efficiency of the oversampling techniques to help in generating a model that correctly classifies instances belonging to the minority class with higher accuracy.

For treating the imbalanced dataset problem, different approaches and methods have been proposed [1, 5–10]. These methods can be classified as undersampling and oversampling methods. The undersampling methods, such as in [1, 10], randomly eliminate instances within the majority class to balance the dataset. But doing so might remove essentials instances from the majority class which leads to decrease in the accuracy of classifying the majority instances against increasing the accuracy of the minority instances classification, whereas the oversampling methods in [6–9] that create new instances within the minority class using systematic algorithms to balance the dataset. This improves the accuracy of minority instances classification without jeopardizing the accuracy of majority instances classification. But it also has some drawbacks such as instance duplication, or over fitting where they might not improve the classification accuracy [5]. Therefore, to ensure generating non-duplicating instances, different heuristics strategies are used while oversampling.

SMOTE [6] is the most used oversampling technique in treating imbalanced datasets. It systematically generates new synthetic instances within the minority region without creating duplicates. SMOTE needs to generate a large number of new instances (e.g., around 500% of the minority instances) [10] to effectively increase the minority class classification accuracy. This is because SMOTE operates on the entire area of the minority region and calculates distances between them for creating new instances. Borderline SMOTE (B-SMOTE) was proposed by Han et al. [9] which generates new instances based on minority borderline instances and other minority instances to calculate the position and values of the new instances, which results in a narrower oversampling area than SMOTE. For that B-SMOTE achieves better classification results with the same or fewer newly generated synthetic instances than SMOTE. However, the number of newly generated instances of B-SMOTE is slightly less than that of SMOTE. Two reasons for that slight difference in the number of new instances and results achieved are as follows: The first is that it is calculated by multiplying the number of minority instances found in the borderline region (80% or more of the minority class instances) by the oversampling percentage, leading to 20% or less in reduction in the number of generated instances. The second reason is that these new instances are generated around the borderline area, and not within the borderline itself, which have the same effect of SMOTE.

1.1 Motivation

There are different approaches to solve the imbalanced dataset weakness such as undersampling and oversampling. While investigating SMOTE, that has different versions and derivatives. Its processes did not cover all its potential parameters such as computing new instances by just using the borderline regions, clustering the borderline, or eliminating small noisy areas of the minority instances that fall within the borderline region. Which motivated us to improve SMOTE, by filling up the gaps, creating different versions and evaluating them. Therefore, a hypothesis was formed to check whether the classifier could get higher classification accuracy for the minority instances if the oversampled instances are generated within the borderline area. The research followed that hypothesis and proposed AB-SMOTE, CAB-SMOTE, and HCAB-SMOTE, to be explained later.

1.2 Paper Contributions

Two different novel oversampling approaches AB-SMOTE and CAB-SMOTE were developed before HCAB-SMOTE. This approach leaded to propose another novel oversampling technique named as HCAB-SMOTE with higher performance.

1.2.1 Affinitive Borderline SMOTE (AB-SMOTE)

It computes and creates the minority danger subset (borderline region) and then oversamples new instances computed within borderline minority instances and increasing its density.

1.2.2 Clustered Affinitive Borderline SMOTE (CAB-SMOTE)

Inspired by the work of [11] that applied *k*-means clustering over all the minority instances and oversampled the clusters. This research developed CAB-SMOTE which computes and creates the minority danger (borderline) subset, applies *k*-means clustering over the minority danger region, and then independently oversamples and generates new instances within each cluster. Therefore, it increases the density within these clusters and improves the classification accuracy of classifying the minority instances more than AB-SMOTE.

1.2.3 Hybrid Clustered Affinitive Borderline SMOTE (HCAB-SMOTE)

It computes and creates the minority danger subset, removes majority borderline instances, applies *k*-means clustering on

the minority borderline region, oversamples clusters larger than 50% of the largest cluster, and removes the smaller clusters.

1.3 Organization of Paper

This paper is organized as follows. Section 2 will have a brief introduction about the related work done in the area of SMOTE, B-SMOTE, and undersampling algorithms. Section 3 will discuss the proposed AB-SMOTE, CAB-SMOTE and HCAB-SMOTE approaches. Section 4 evaluates the methodology and mentions which software and datasets were used, displays the results, compares and analyzes them. Finally, Sect. 5 provides the conclusion and discusses future work. Table 1 shows all the acronyms used and their definitions to make them clearer and easier to find.

2 Background

Different preprocessing techniques were used to solve the imbalanced data set problem [5]. This section mention about over- and undersampling techniques that take place before the classification stage in the data mining process to change the data distribution. The oversampling generates more instances within the minority class, whereas the undersampling deletes instances falling the majority class in order to balance the dataset, thereby allowing the classifier algorithm to act in an unbiased way.

2.1 Oversampling

2.1.1 Synthetic Minority Oversampling Technique (SMOTE)

SMOTE, the most used oversampling technique in machine learning, was proposed by [6] and has input parameters that the user can control and change. These parameters are as follows: K the number of nearest neighbors, with a default value of 5, meaning that the algorithm uses 5 nearest neighbors around the selected instance to pick from, it also has the oversampling percentage parameter with value of 100% by default, which orders the algorithm to generate 100% more of minority instances, hence doubling the number of minority instances, and it also contains a random seed number that has a value of 1 as default which is used for experimental purposes to get the same random number in each run so as to compare the results. SMOTE loads the dataset to the memory, defines the minority class by counting and comparing the number of instances in each class, and chooses the one that has the least number of instances. After that, it chooses one minority instance and calculates its k neighbors. After that, it randomly chooses one of k nearest neighbors to calculate a distance vector, which is the difference between the initial minority instances and the other selected instances from the same class. After computing the distance, it will be multiplied with a random gap number, having a value between 0 and 1 to pick a point on the line from initial minority instance to generate the new instance. Then SMOTE continues the same process with other minority instances until it reaches the percentage value given by

Table 1 List of acronyms and their definitions

Acronyms	Definitions
SMOTE	Synthetic Minority Oversampling Technique
B-SMOTE	Borderline SMOTE
AB-SMOTE	Affinitive Borderline SMOTE
CAB-SMOTE	Clustered Affinitive Borderline SMOTE
HCAB-SMOTE	Hybrid Clustered Affinitive Borderline SMOTE
I.R.	Imbalanced ratio
nn	Nearest neighbors
Sample; P_i	Minority instance: $P_1, P_2 \ldots P_{num}$
N_i	Majority instance: $N_1, N_2 \ldots N_{num}$
$S\%$	Input: oversampling percentage value to generate
S	Newly generated instance
Ns	Number of samples to generate
Danger; P'_i	Borderline instance: $P'_1, P'_2 \ldots P'_{num}$
KNN	K nearest neighbors: used in the oversampling stage to generate new instances
Mnn	M nearest neighbors: used to compute the borderline instances
M'	Number of majority instances found within the Mnn from the minority instance at different iterations
Dif	Difference between two instances
k-means	Number of clusters which the data will be clustered to

the user. Figure 1 shows 8 iterations of the algorithm, where 8 new instances belonging to the positive minority class are generated. This is done using Eq. (1) to calculate the new values for the newly generated instances where P_i is the instance that the algorithm chose and computed the KNN around it, and P_j is the randomly picked instance from the KNN.

$$\text{New instance} = P_i + \text{gap} * (\text{distance}(P_i, P_j)) \quad (1)$$

Figure 2 displays the pseudo code of SMOTE, where functions were created for each process to be used for the other algorithm in an understandable way. These functions are gathered in Table 2.

SMOTE generates randomly new instances all over the dataset, leading to an increase in noise within the majority class area, or within the safe minority region far from the borderline area and over fitting it, therefore not efficiently increasing the classification accuracy for classifying the minority instances. For these reasons, SMOTE has different derivatives such as Safe-level-SMOTE [7], SMOTEBOOST [8], Borderline SMOTE [9] and others, which were proposed to overcome or minimize these problems. This research focused on developing and enhancing B-SMOTE.

2.1.2 Borderline SMOTE

SMOTE had been developed and different derivatives came out from its development; one of them is B-SMOTE that was proposed in [9] to extend SMOTE by differentiating between the minority instances into danger, safe and noise instances by counting the number of majority instances M' within M nearest neighbors (Mnn) with default value of 5, between each minority instances P_i and other instances from the dataset. The instance is considered to be safe when the number of majority examples M' among the M nearest neighbors is within the interval 0 to $M/2$; in other words, when most of the instances around the chosen minority instance are from the minority class. The instance is considered to be as noise when all the M nearest neighbors of intended instance are majority examples $M' = M$. Safe and noise instances were eliminated from the computation function to narrow the oversampling area around the border. The instance is considered to be a danger instances P' when the number of majority instances M' that are found within the Mnn is between $M/2$ and M and thus added to a new subset called danger. After that, B-SMOTE measures KNN between borderline instances (danger) and other minority instances, which might also be within the borderline region, then generates the new instance using Eq. (2):

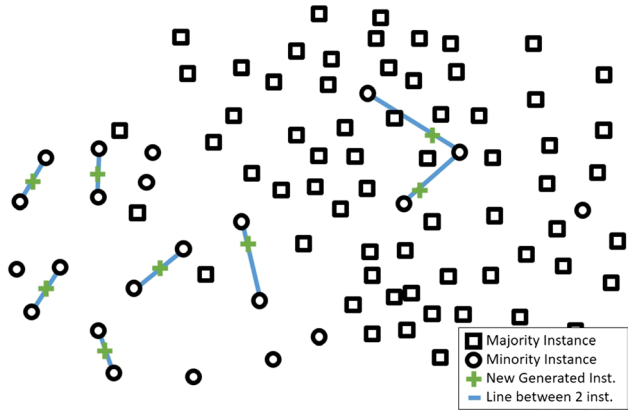

Fig. 1 SMOTE diagram

Algorithm 1:SMOTE

Input: P number of minority class sample ; S% amount of synthetic to be generated; k Number of nearest neighbors
Output: Ns= (S/100)* P synthetic samples

1. ***Create function ComputKNN (i←1 to P, P_i,P_j)***
   { **For** i ← 1 to P
   Compute k nearest neighbors of each minority instance P_i and other minority instances P_j.
   Save the indices in the nnarray.
   Populate (Ns, i, nnarray) to generate new instance.
   End for }
2. Ns= (S/100)*P
   While Ns ≠ 0
3. ***Create function GenerateS (***P_i, P_j)
   {Choose a random number between 1 and k, call it nn.
   For attr ← 1 to numattrs
   dif=P_i [nnarray[nn]][attr] – P_j [i][attr]
   gap = random number between 0 and 1
   Synthetic [newindex][attr] = P_i[i][attr] + gap ∗ dif
   End for
   newindex= newindex +1}
   Ns = Ns – 1
   End while
4. Return (∗ End of Populate. ∗)

End of Pseudo-Code.

Fig. 2 SMOTE algorithm

Table 2 List of functions and their definitions

Function	Definitions
ComputKNN	($i \leftarrow 1$ to P, P_i, P_j) Compute the KNN between P & P
	($i \leftarrow 1$ to P', P'_i, P_j) Compute the KNN between P' & P
	($i \leftarrow 1$ to P', P'_i, P'_j) Compute the KNN between P' & P'
GenerateS	(P, P) Create instance & fill its values between P & P
	(P'_i, P_j) Create instance & fill its values between P' & P
	(P'_i, P'_j) Create instance & fill its values between P' & P'
MinDanger	Creates the minority border P' subset

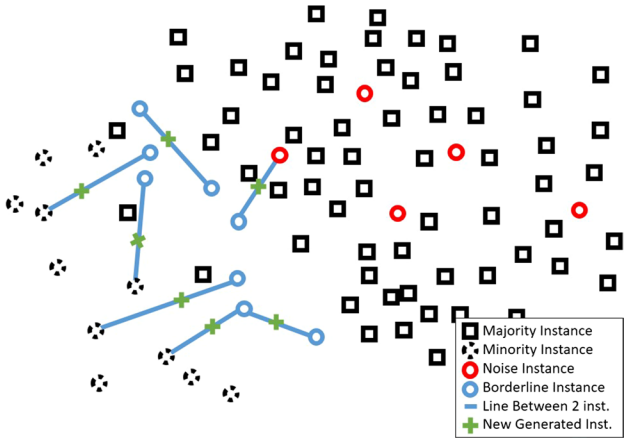

Fig. 3 Borderline SMOTE diagram

Algorithm 2:B-SMOTE

Input: P number of minority class sample ; S% amount of synthetic to be generated; M number of nearest neighbors to create the borderline subset; k Number of nearest neighbors

Output: (S/100)* P' synthetic samples

1. ***Create function MinDanger ( )***
   { **For** i ← 1 to P
   Compute M nearest neighbors of each minority instance and other instances from the dataset.
   Check number of Majority instance M' within the Mnn
   IF M/2< M' < M
   | Add instance P to borderline subset P'
   End if
   End for }
2. **ComputKNN (i←1 to P', P_i',P_j)**
3. Ns= (S/100)*P'
   While Ns ≠ 0
4. ***GenerateS (***P'$_i$, P$_j$**)**
   Ns = Ns – 1
   End while
5. Return (∗ End of Populate. ∗)

End of Pseudo-Code.

Fig. 4 Borderline SMOTE algorithm

$$\text{New instance} = P'_i + \text{gap} * (\text{distance}\,(P'_i, P_j)) \tag{2}$$

where P'_i is the borderline minority instance, P_j is the random chosen KNN minority instance, and the gap is a random number having a value between 0 and 1. The danger instances are displayed as the blue-colored circles in Fig. 3, whereas the B-SMOTE algorithm is demonstrated in Fig. 4.

Another version of B-SMOTE was also proposed by Han et al. [9] and called B-SMOTE 2 where it uses instances from the majority class in the KNN calculation. But while it uses a majority instance, the gap values used will be between 0 and 0.5 so that the newly created instances will be created closer to the borderline instance. B-SMOTE oversampling is focused around the borderline region, thus widening the region and might confuse the classifier.

2.2 Undersampling Methods

Another imbalanced dataset treatment is by cleaning the data with undersampling, where instances from the majority class are removed from the data in order to balance the class distribution [10, 12, 13]. The undersampling methods can be separated into two types: random undersampling where it randomly removes majority instances without using selective criteria; and focused undersampling that removes majority instances located on the borders between the two classes [14]. The undersampling methods have their advantages and disadvantages. They are beneficial in eliminating a large number of majority instances that significantly reduces the size of the datasets, and decreases the run time. However, while doing that, the classifier might lose important information that contribute to the learning process leading to a decrease in the majority instance classification accuracy [15]. Discusses different methods used in the undersampling process such as Tomek Link and Nearest neighbors [16] to identify the majority instances to be eliminated from the dataset. Tomek link is defined as a pair of instances where x is an instance of class A and y is an instance of class B, and the distance between x and y is denoted by $d(x, y)$. The points (x, y) are considered a T-Link, if for any instance z where, $d(x, y) < d(x, z)$ or $d(x, y) < d(y, z)$. If any two instances are identified as T-Link, then one of them is considered noise and the instance having the majority class will be removed.

3 Proposed Approaches

First, this research extended B-SMOTE to AB-SMOTE to focus the generated instances within the borderline area, second AB-SMOTE developed to CAB-SMOTE to cluster the borderline and oversample within each cluster. Finally, the research developed HCAB-SMOTE that uses undersampling technique to remove noisy borderline instances from the majority class and oversamples within the minority class clustered borderline area to get the best results. The source codes are uploaded to GITHUB website under this URL github.com/Hishammajzoub/HCAB-SMOTE.

3.1 Affinitive Borderline SMOTE

Since the WEKA software for data mining does not have B-SMOTE oversampling filter. The open source code of SMOTE was obtained, understood how it functions, and extended it to B-SMOTE. While doing so, some gaps which B-SMOTE did not tackle were noticed. One of the gaps was that the new instances were generated around the borderline

area and not inside it which motivated to the enhancement for the B-SMOTE to increases the density of the minority borderline instances, generates new instances within the borderline area, and called it Affinitive Borderline SMOTE (AB-SMOTE). That has similar computation process to create danger subset as that of B-SMOTE, but different KNN functionality, where it locks the KNN within border region instances, without using instances outside the border area for the oversampling, thus limiting the sample diversity used for generating new instances to the border region. AB-SMOTE is displayed in Fig. 5 and can be compared with Fig. 3 to check the differences between the two algorithms, whereas Fig. 6 describes the AB-SMOTE pseudo code, where it computes for every minority instance its M nearest neighbors within the whole training data to create the danger subset. Later the algorithm checks the KNN between each danger instance and other instances within the same danger subset, randomly chooses one of the KNN, calculates the distance between the two instances and applies Eq. (3) for computing the value of the new instance.

$$\text{New instance} = P'_i + \text{gap} * (\text{distance}(P'_i, P'_j)) \tag{3}$$

3.2 Clustered AB-SMOTE

Unlike existing approach such as the work in [11] that applies *k*-means clustering over all the instances of the minority class, CAB-SMOTE applied the clustering process only on the minority borderline instances, thus dividing the danger subset into different clusters. Each cluster will have different number of instances, so it was possible to compare their sizes to be used for different criteria. The oversampling function is applied within each cluster, narrowing the oversampling region into each cluster inside the borderline area. Different oversampling and stopping strategies were used, and the classification accuracies were recorded and compared to choose the best one. All the strategies undergo the same process for building the danger subset, and *k*-means clustering. The *k*-means clustering values used were 2, 3, 5, and 7 applied on danger subset. These values were used to have a clear observation for experiment, where *k*-means = 2 which is the minimum value for clustering process to function, and then the values of 3, 5 and 7, were used to divide the borderline area into more clusters. Increasing the clustering to more than 7 clusters might reach a point where it would not be useful and increases the computation costs. That is why *k*-means stopped at value of 7. Then applied oversampling on the clusters using KNN of the instances found within each cluster as shown in Figs. 7 and 9.

The different oversampling strategies work as follows:

Algorithm 3:AB-SMOTE

Input: P number of minority class sample ; S% amount of synthetic to be generated; M number of nearest neighbors to create the borderline subset; k Number of nearest neighbors
Output: (S/100)* P' synthetic samples
1. **MinDanger ( )** to create danger subset
2. **ComputKNN (i←1 to P', P_i',P_j')**
3. Ns= (S/100)*P'
   While Ns ≠ 0
4. ***GenerateS (***P'$_i$, P$_j$'**)**
   Ns = Ns – 1
   End while
5. Return (∗ End of Populate. ∗)
End of Pseudo-Code.

Fig. 6 Affinitive Borderline SMOTE algorithm

3.2.1 CAB-SMOTE Strategy I

The algorithm creates the borderline subset using Mnn, then clusters that subset using *k*-means values, counts the

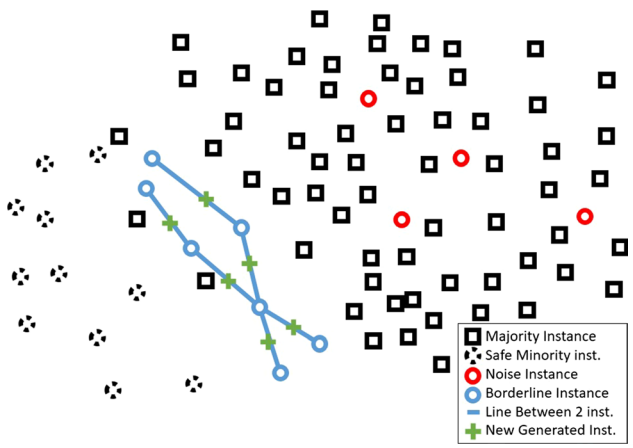

Fig. 5 Affinitive Borderline-SMOTE diagram

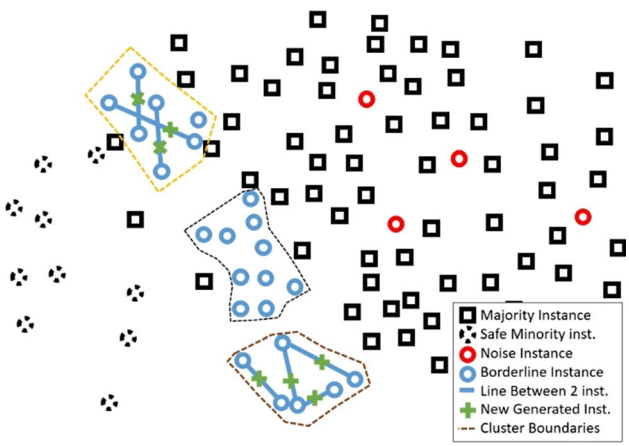

Fig. 7 CAB-SMOTE strategy I diagram

instances within each cluster, holds the largest cluster, and oversamples the other clusters till they reach the size of the largest cluster.

Figure 7 shows how this strategy works by clustering the danger area into three clusters, having 7, 10, and 6 instances, respectively. The algorithm added 3 new instances for the first cluster, kept the second cluster unchanged, and added 4 new instances for the third cluster which made all the clusters have equal number of instances. Figure 8 shows the pseudo code this algorithm.

3.2.2 CAB-SMOTE Strategy II

CAB-SMOTE SII creates the borderline subset using Mnn, then clusters that subset using *k*-means values, counts the instances within each cluster, checks the number of instances in the largest cluster, and oversamples all the clusters larger than half of the largest cluster including the largest cluster as shown in Figs. 9 and 10. Here, the stopping criteria are the input percentage with default value of 100% (doubling the instances in clusters larger than half of the largest cluster).

3.2.3 CAB-SMOTE Strategy III

CAB-SMOTE SIII creates the borderline subset using Mnn, then clusters that subset using *k*-means values, oversamples all clusters without any conditional criteria till it reaches the given oversampling percentage. It is developed this way to be compared with other strategies to check the effect of small clusters in the danger area on *F*-measure values.

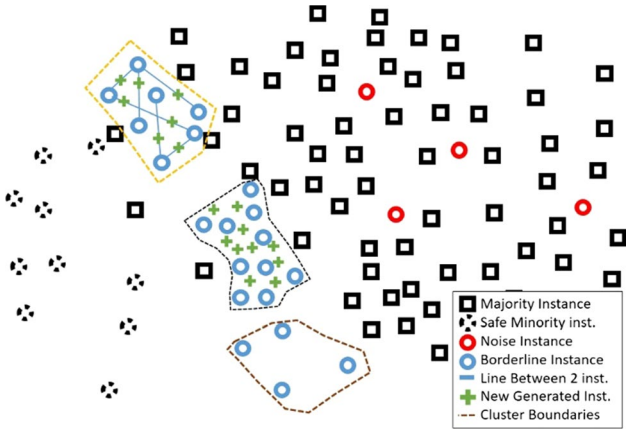

Fig. 9 CAB-SMOTE strategy II diagram

3.3 Hybrid CAB-SMOTE

This approach is developed to clean the dataset from some noisy instances, such as the negative majority borderline instances and small minority clusters within the borderline. It copies all the needed instances and extracts them to a new dataset to be used for model building and classification. Figures 11 and 12 display how HCAB-SMOTE work.

The subsets created within this method are displayed below.

Algorithm 4:CAB-SMOTE Strategy1

Input: P number of minority class sample ; S% amount of synthetic to be generated; M number of nearest neighbors to create the borderline subset; k Number of nearest neighbors
Output: (S/100)* P' synthetic samples
1. **MinDanger ( )** to create danger subset
   Apply k-means clustering on P' subset
   Nmax: number of instances found in the largest cluster
2. **For** km ← 0 to k-means
   Ns=0
3. Ns= Nmax - number of instance in cluster km
   While Ns ≠ 0
4. **ComputKNN (i←1 to P', P_i',P_j')**
5. ***GenerateS (***P'$_i$, P$_j$')
   Ns = Ns – 1
   End while
   End for
6. Return (∗ End of Populate. ∗)
End of Pseudo-Code.

Fig. 8 CAB-SMOTE strategy I algorithm

Algorithm 5:CAB-SMOTE Strategy 2

Input: P number of minority class sample ; S% amount of synthetic to be generated; M number of nearest neighbors to create the borderline subset; k Number of nearest neighbors
Output: (S/100)* P' synthetic samples
1. **MinDanger ( )** to create danger subset
   Apply k-means clustering on P' subset
   Nmax: number of instances found in the largest cluster
2. **For** km ← 0 to k-means
   Ns=0
3. **If** number of instance in cluster km >= 0.5* Nmax
   Ns= (S/100)*P'
   While Ns ≠ 0
4. **ComputKNN (i←1 to P', P_i',P_j')**
5. ***GenerateS (***P'$_i$, P$_j$')
   Ns = Ns – 1
   End while
   End if
   End for
6. Return (∗ End of Populate. ∗)
End of Pseudo-Code.

Fig. 10 CAB-SMOTE strategy II

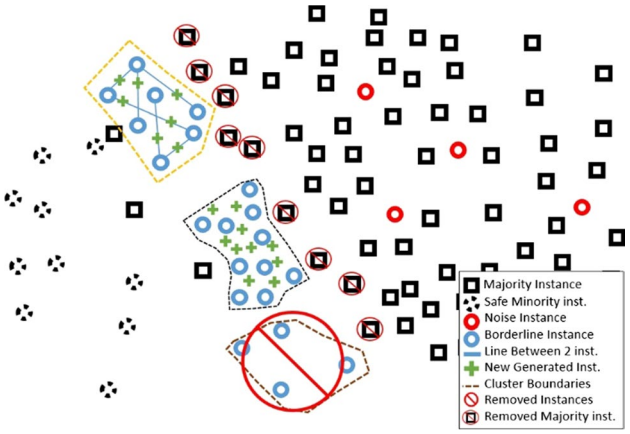

Fig. 11 HCAB-SMOTE diagram

Algorithm 6:HCAB-SMOTE

Input: P number of minority class sample ; S% amount of synthetic to be generated; M number of nearest neighbors to create the borderline subset; k Number of nearest neighbors; k-means value

Output: (S/100)* P' synthetic samples

1. **Call MinDanger** to create danger subset
   Add all majority instances to Majnondanger arrey
2. **For** i ← 1 to N
   Compute M nearest neighbors for each majority instance Ni and other instances from the dataset.
   Check number of Minority instance P within the Mnn
   If M/2< P < M
   remove instance Ni from Majnondanger
   End if
   End for
   Apply k-means clustering on MinDanger P' subset
   Nmax: number of instances found in the largest cluster
3. **For** km ← 0 to k-means
   Ns=0
4. **If** number of instance in cluster km >= 0.5* Nmax
   Copy km cluster to Newtotal subset
   Ns= (S/100)*P'
   While Ns ≠ 0
5. **Call ComputKNN (i←1 to P', P_i',P_j')**
6. ***GenerateS (***P'$_i$, P$_j$'**)**
   Ns = Ns – 1
   End while
   End if
   End for
7. Copy Majnondanger, Minoritynondanger, and Generated subsets to Newtotalsubset.
8. Export Newtotalsubset

End of Pseudo-Code

Fig. 12 HCAB-SMOTE approach

- *Total* holds all the instances and computes the nn
- *Minority* holds all minority instances *P*
- *MinDanger* consists of minority border instances *P′*
- *Minoritynondanger* holds minority non border instances
- *Majority* holds all majority instances
- *Majnondanger* holds non-border majority instances
- *Generated* holds oversampled new synthetic instances S
- *Newtotal* consists of non-noisy minority borderline, Minoritynondanger, Majnondanger, and Generated instances.

The *Minority* and *Majority* subsets are created by checking the class id value of each instance by using the procedure of the original SMOTE for classifying the minority instances. The *MinDanger* subset is created by checking the *M* nearest neighbor (Mnn) between each minority instance and all instances in the dataset (Total), if *M* has a value of 8, the algorithm checks within the 8 closest instances around that minority instance and counts the number of majority instances (*M′*), if the number of *M′* is between M/2 and M then this instance is considered a minority borderline instance and is added to *MinDanger* subset. In contrast for the *Majnondanger* subset which tends to clean the dataset from borderline majority instances, the algorithm first copies all the majority instances into *Majnondanger* subset, checks the Mnn between each majority instance Ni and all other instances, and counts the number of minority instances found within the Mnn; if the number of minority instances falls between *M*/2 and *M*, the algorithm deletes that instance Ni from the *Majnondanger* subset. After that, the algorithm applies *k*-means clustering on the *MinDanger* subset, separating it into *k* clusters, counts the number of instances within each cluster, and saves the number of instances found in the largest cluster. Then it compares the number of instances in each cluster to the number of instances inside the largest one, if a cluster has more than half of the instances of largest cluster, the instances within this cluster are copied to the *Newtotal* subset, and oversampling process is applied on that cluster to create the new instances saved to *Generated* subset.

After the oversampling process finishes, the *Majnondanger*, the *Minoritynondanger* and the *Generated* subsets are copied to the *Newtotal* subset, which is exported to a new arff dataset file. At this point, the HCAB-SMOTE process is completed, and the new synthesized arff dataset is then loaded into the program to apply the classification algorithm for model building and evaluation. HCAB-SMOTE extracts a new dataset because it was not possible to directly remove the negative borderline instances from the main dataset.

4 Oversampling Techniques Evaluation

In order to evaluate the oversampling approaches, WEKA [17] is used, which is an open source java script software for data mining which enabled reading, and modifying of its code where it went under investigation in its oversampling

filters such as SMOTE and developed new derivatives from it, such as B-SMOTE, AB-SMOTE, CAB-SMOTE, and HCAB-SMOTE. In order to test and evaluate the oversampling methods, different supervised binary real-life datasets were used. These datasets are displayed in Table 3 and obtained from different online repositories which are widely used in machine learning scientific researches. These datasets were downloaded from Crowd analytics, UC Irvine Machine Learning Repository, Keel, and IBM. This research used the same criteria considered in [18] for choosing the imbalanced datasets, where the number of minority instances should be less than 40% of the number of instances of the majority class to be considered as an imbalanced dataset. The imbalanced ratios (I.R.) found in Table 3 are calculated by dividing the number of minority instances over the number of majority instances. The datasets varied between abalone sea shells age prediction dataset [19], two Customer churn datasets [20, 21] that recorded historical data about customers and flagged the churned ones that unsubscribed from the service, in order to create a model to predict the customers who might churn in the future to develop customer retention programs to keep those customers from churning. Haberman's Survival dataset [22] which consists of recorded data from a study conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on patients who had undergone surgery for breast cancer and noted if they died or survived within 5 years after the surgery [23]. Employee Attrition dataset [24] for tracking and analyzing employee satisfaction for retaining valuable employees. Solar Flare dataset [25], Wine quality dataset [4], Sales dataset [26], Bank dataset [27] which was related to banking direct phone marketing campaigns of a Portuguese banking institution to ask clients for subscribing to bank term deposit. An Adult Census income dataset [28] for determining if the person makes over $50k per year. Amazon Employee access dataset [29] which used to classify which new employees would have a denied access which stop them from fulfilling their role. And a Click Prediction dataset [30] to predict if a user will click on a given advertisement.

Table 4 Oversampling techniques used in the experiment

Algorithm	Mnn	KNN	GaP	*k*-means	Per.
1.0: Original	–	–	–	–	–
2.x: SMOTE 2V	–	5, 8	0–1	–	100
3.x: B-SMOTE *P′.P* 2V	5, 8	5, 8	0–1	–	100
3.3: B-SMOTE SII *P′*.N 2V	5, 8	5, 8	0–0.5	–	100
4.x: AB-SMOTE *P′.P′* 2V	5, 8	5, 8	0–1	–	100
5.x: CAB-SMOTE SI 8V	5, 8	5, 8	0–1	2, 3, 5, 7	–
6.x: CAB-SMOTE SII 8V	5, 8	5, 8	0–1	2, 3, 5, 7	100
7.x: CAB-SMOTE SIII 8 V	5, 8	5, 8	0–1	2, 3, 5, 7	100
8.x: HCAB-SMOTE 8 V	5, 8	5, 8	0–1	2, 3, 5, 7	100

The first step was to classify the datasets using the decision tree classification algorithm, called J48 in WEKA. The classification was applied using *k*-fold cross-validation with $k=5$ which divides the dataset into five folds, maintaining the same proportion in class distribution over the dataset. In other words, if the whole dataset had 1000 instances consisting of 800 majority instances and 200 minority instance, while using fivefold cross-validation, this dataset would be divided into 5 different subsets, each of them would have 200 instances consisting of 160 majority instances and 40 minority instances, thus keeping the same distribution. Then the classification algorithm is applied for five iterations. In each iteration, it will use four folds for training and holds one fold for testing and saves the results, and for the other iterations, it will hold a different testing fold and train on the other folds, and save the results of each iteration. After last-fold cross-validation is done, it will print out the average of those results into WEKA results window [31, 32]. After recording the results, different oversampling techniques with different tuning parameters were applied using fivefold cross-validation to compare the results as shown in

Table 3 Datasets used in the research

Datasets	Total	Minority	Majority	I.R. ratio (%)	Source
Abalone9-18	731	42	689	6	[19]
Customer Churn Wireless Telecom	5000	707	4293	16.4	[20]
Telco customer churn	7043	1869	5174	36.12	[21]
Haberman's survival	306	81	225	36	[22]
Employe attrition	1470	237	1233	19.2	[24]
Flare	1109	43	1066	4	[25]
Wine quality	1493	53	1546	3.4	[4]
Sales: win–loss	78,025	17,627	60,398	29.18	[26]
Bank	45,211	5289	39,922	13.24	[27]
Adult census income	32,561	7841	24,720	31.71	[28]
Amazon employee access	32,769	1897	30,872	6.14	[29]
Click prediction	39,948	6728	33,220	20.25	[30]

Table 4 where P denotes to the positive minority instances, P' denotes the minority borderline instances, N denotes the majority instance, Gap is the number which is multiplied by the distance between the two selected instances to pinpoint the location of where the new instance would be located within the line connecting both instances and assign that value to the new instance, Mnn between minority instances and the whole dataset for defining the borderline subset, KNN that is used between the chosen instances in the final stage of oversampling, k-means is the number of clusters that the danger area will be divided into, and Per. stands for percentage which is the oversampling percentage value of how many new instances should be generated, SI, SII, and SIII stand for different strategies used for CAB-SMOTE, and V. is the number of versions of the oversampling technique, while each version has different KNN and Mnn input parameters. The first version used KNN and Mnn equal 5, while the second used KNN and Mnn equal to 8. These two versions were applied for all the techniques, but regarding CAB-SMOTE and HCAB-SMOTE they were repeated for each k-means values (2, 3, 5 and 7). After getting the results, the Recall and F-measure values for classifying the minority class (explained in the next section) for each dataset were recorded and the version that gave the highest F-measure value within each oversampling approach was chosen in order to compare them. Then applied oversampling over those versions with different percentages such as 50, 100, 200, 300, and 400 to observe the progress of the oversampling methods, which will be shown in the results section. However, the algorithms B-SMOTE SII 3.3 and 3.4 that use minority borderline instances and majority instances for the oversampling process did not give better results than B-SMOTE 3.1 and 3.2; therefore, they were not included in the oversampling with different increasing percentages.

To get an accurate evaluation of the classifier applied on the imbalanced dataset, the classification accuracy is not enough, because it does not reflect the true classifier accuracy in predicting the minority class. Thus, a confusion matrix such displayed in Table 5 is used for monitoring different metrics to get a more detailed image about the results which deliver better evaluation of the classifier predicting power for the minority class [33]. In imbalanced datasets, the majority class is considered as negative, whereas the minority class is positive. For the classifier to have a perfect predicting accuracy regarding the minority instances in the confusion matrix, the true positive (TP) number should be equal to the total number of the minority samples, which implies that the model is correctly predicting all actual positive instances to be in a positive class. Since supervised datasets have a defined class for each instance, the prediction accuracy by comparing the predicted class for instances with their actual class is possible.

Confusion Matrix definitions:

- *TP/True Positive* The classifier is predicted positive, while the actual value is positive.
- *FP/False Positive* The classifier is predicted positive, but the actual value is negative (type 1 error).
- *FN/False Negative* The classifier is predicted negative, but the actual value is positive (type 2 error).
- *TN/True Negative* The classifier is predicted negative, and the actual value is also negative.
- *Recall* shows the ratio of how much the classifier correctly classified a minority instance from all actual positive instances.
- *P.P.V./Positive Prediction Value (Precision)* It shows how much the classifier correctly classified a minority instance from all predicted positive instances.

The classifiers were evaluated with recall and F-measure that show up as an output in WEKA after applying the decision tree classifier with fivefold cross-validation.

1. F-measure: is a measurement that presents the harmonic mean of recall and precision [33], and F-measure is calculated with Eq. (4).

$$F\text{ - measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

To achieve maximum accuracy, the values of recall and F-measure should be very close to 1. If these values were equal to 1, it implies that the model is correctly predicting all the actual minority class instances to be in the minority class and all the actual negative class instances to be in majority class.

AB-SMOTE was evaluated against B-SMOTE, where they generated the same number of instances based on calculations between the actual minority instances. The results showed that AB-SMOTE got higher recall and F-measure than B-SMOTE within most of the datasets, and mainly the ones that have large number of instances as shown in Table 6. This was the motivation to develop CAB-SMOTE.

Table 5 Confusion matrix

Predicted values	Actual values			
	Positive	Negative		
Positive	TP	FP	P.P.V.	$\frac{TP}{(TP+FP)}$
Negative	FN	TN	N.P.V	$\frac{TN}{(FN+TN)}$
	Recall $\frac{TP}{(TP+FN)}$	Specificity $\frac{TN}{(TN+TP)}$	Accuracy $\frac{(TP+TN)}{(TP+FP+TN+FN)}$	

Table 6 B-SMOTE and AB-SMOTE accuracy comparison

Dataset	Recall		*F*-measure	
	B-SMOTE	AB-SMOTE	B-SMOTE	AB-SMOTE
Abalone	X		X	
Churn	X		X	
Haberman		X	X	
Employee Attrition		X		X
Telco Customer churn		X		X
Flare	X		X	
Wine Quality	X	X	X	X
Sales, win loss		X		X
Bank Subscription		X		X
Adult income		X		X
Amazon Employee		X		X
Click Prediction		X		X
Total	4	9	5	8

Table 7 Results from Abalone Dataset

Maj. = 689; Min. = 42	Border M5 = 40–95.2% Border M8 = 39–92.8%			I.R. = 6.1%
Oversampling technique	*B*%	Gen	Recall	*F*-measure
1.0: Original	0	0	0.167	0.241
2.1: SMOTE: 8	0	42	0.417	0.486
3.1: B-SMOTE P' & P: 8, 8	100	39	0.395	0.496
4.1: AB-SMOTE P' & P': 8, 8	100	39	0.37	0.438
5.4: CAB-SMOTE SI: 5, 5, 5	100	40	0.427	0.493
6.0: CAB-SMOTE SII: 5, 5, 2	100	40	0.549	0.629
7.0: CAB-SMOTE SIII: 5, 5, 2	100	40	0.549	0.629
8.1: HCAB-SMOTE: 8, 8, 2	100	39	**0.617**	**0.699**

Table 8 Results from Customer Churn Dataset

Maj. = 4293; Min. = 707	Border M5 = 534–75% Border M8 = 517–73%			I.R. = 16.47%
Oversampling technique	*B*%	Gen	Recall	*F*-measure
1.0: Original	0	0	0.639	0.748
2.0: SMOTE 8	0	707	0.631	0.755
3.1: B-SMOTE P' & P: 8, 8	100	517	0.595	0.718
4.1: AB-SMOTE P' & P': 8, 8	100	517	0.585	0.711
5.3: CAB-SMOTE SI: 8, 8, 3	67	362	0.77	0.838
6.0: CAB-SMOTE SII: 5, 5, 2	55	298	0.787	0.855
7.0: CAB-SMOTE SIII: 5, 5, 2	100	534	0.816	0.867
8.2: HCAB-SMOTE: 5, 5, 3	80	432	**0.822**	**0.879**

Table 9 Results from Haberman Dataset

Maj. = 225; Min. = 81	Border M5 = 65–80.2% Border M8 = 61–75.3%			I.R. = 36%
Oversampling technique	B%	Gen	Recall	*F*-measure
1.0: Original	0	0	0.272	0.331
2.0: SMOTE 5	0	81	0.617	0.643
3.0: B-SMOTE P' & P:5, 5	100	65	0.541	0.583
4.1: AB-SMOTE P' & P': 8, 8	100	61	0.57	0.581
5.5 CAB-SMOTE SI: 8, 8, 5	121	74	0.69	0.656
6.6: CAB-SMOTE SII: 5, 5, 7	78	51	0.712	0.651
7.5: CAB-SMOTE SIII: 5, 5, 3	100	61	**0.775**	0.673
8.6: HCAB-SMOTE: 5, 5, 7	78	51	0.729	**0.761**

Table 10 Results from Employee Attrition Dataset

Maj. = 1233; Min. = 237	Border at M5 = 236–99% Border M8 = 232–97.8%			I.R. = 19.2%
Oversampling technique	B%	Gen	Recall	*F*-measure
1.0: Original	0	0	0.181	0.265
2.0: SMOTE 8	0	236	0.501	0.59
3.0: B-SMOTE P' & P: 5, 5	100	236	0.501	0.59
4.0: AB-SMOTE P' & P': 5, 5	100	236	0.516	0.594
5.6: CAB-SMOTE SI: 5, 5, 7	89	212	0.528	0.605
6.0: CAB-SMOTE SII: 5, 5, 2	100	236	0.588	0.673
7.0: CAB-SMOTE SIII: 5, 5, 2	100	236	0.588	0.673
8.6: HCAB-SMOTE: 5, 5, 7	57	135	**0.605**	**0.696**

Table 11 Results from Telco Customer Churn Dataset

Maj. = 5174; Min. = 1869	Border M5 = 1101–58.9% Border M8 = 949–50.7%			I.R. = 36.12%
Oversampling technique	B%	Gen	Recall	*F*-measure
1.0: Original	0	0	0.489	0.543
2.0: SMOTE 8	0	1869	0.923	0.732
3.0: B-SMOTE *P′* & *P*: 5, 5	100	1101	0.895	0.677
4.0: AB-SMOTE *P′* & *P′*: 5, 5	100	1101	0.897	0.678
5.4: CAB-SMOTE SI: 5, 5, 5	114	1259	**0.932**	0.705
6.6: CAB-SMOTE SII: 5, 5, 7	99	943	0.811	0.718
7.6: CAB-SMOTE SIII: 5, 5, 7	100	1101	0.918	0.699
8.6: HCAB-SMOTE: 5, 5, 7	99	943	0.876	**0.808**

Table 12 Results from Flare Dataset

Maj. = 1066; Min. = 43	Border M5 = 39–90.6% Border M8 = 41–95.3%			I.R. = 4.3%
Oversampling technique	B%	Gen	Recall	*F*-measure
1.0: Original	0	0	0	0
2.0: SMOTE 5	0	43	0.36	0.47
3.1: B-SMOTE *P′* & *P*: 8, 8	100	41	0.369	0.434
4.0: AB-SMOTE *P′* & *P′*: 5, 5	100	39	0.293	0.421
5.1: CAB-SMOTE SI: 8, 8, 2	51	21	0.375	0.5
6.0: CAB-SMOTE SII: 5, 5, 2	100	39	0.488	0.615
7.1: CAB-SMOTE SIII: 8, 8, 2	100	41	0.56	0.618
8.0: HCAB-SMOTE: 5, 5, 2	100	39	**0.61**	**0.709**

Table 13 Results from Wine Quality Dataset

Maj. = 1546; Min. = 53	Border M5 = 53–100% Border M8 = 53–100%			I.R. = 3.4%
Oversampling technique	B%	Gen	Recall	*F*-measure
1.0: Original	0	0	0	0
2.0: SMOTE 5	100	53	0.151	0.215
3.0: B-SMOTE *P′* & *P*: 5, 5	100	53	0.151	0.215
4.0: AB-SMOTE *P′* & *P′*: 5, 5	100	53	0.151	0.215
5.2: CAB-SMOTE SI: 5, 5, 3	98	52	0.286	0.38
6.3: CAB-SMOTE SII: 5, 5, 3	66	35	0.341	0.458
7.0: CAB-SMOTE SIII: 5, 5, 2	100	53	0.415	0.442
8.3: HCAB-SMOTE: 8, 8, 3	66	35	**0.529**	**0.643**

Table 14 Results from sales win–loss dataset

Maj. = 60,398; Min. = 17,627	Border M5 = 15,176–86% Border M8 = 14,620–82.9%			I.R. = 29.2%
Oversampling technique	B%	Gen	Recall	*F*-measure
1.0: Original	0	0	0.644	0.695
2.0: SMOTE 5	0	17,627	0.816	0.838
3.0: B-SMOTE *P′* & *P*: 5, 5	100	15,176	0.799	0.817
4.0: AB-SMOTE *P′* & *P′*: 5, 5	100	15,176	0.804	0.826
5.7: CAB-SMOTE SI: 8, 8, 7	99	14,618	0.806	0.839
6.2: CAB-SMOTE SII: 5, 5, 3	100	15,176	0.808	0.841
7.0: CAB-SMOTE SIII: 5, 5, 2	100	15,176	0.807	0.841
8.0: HCAB-SMOTE: 5, 5, 2	100	15,176	**0.844**	**0.864**

Table 15 Results from bank subscription dataset

Maj. = 39,922; Min. = 5289	Border M5 = 4331–81.8% Border M8 = 3896–73.6%			I.R. = 13.25
Oversampling technique	*B%*	Gen	Recall	*F*-measure
1.0: Original	0	0	0.49	0.546
2.1: SMOTE 8	0	5289	0.759	0.737
3.0: B-SMOTE *P′* & *P*: 5, 5	100	4331	0.684	0.693
4.0: AB-SMOTE *P′* & *P′*: 5, 5	100	4331	0.685	0.703
5.6: CAB-SMOTE SI: 5, 5, 7	89	3873	0.696	0.743
6.0: CAB-SMOTE SII: 5, 5, 2	100	4331	0.708	0.756
7.0: CAB-SMOTE SIII: 5, 5, 2	100	4331	0.708	0.756
8.0: HCAB-SMOTE: 5, 5, 2	100	4331	**0.796**	**0.821**

Table 16 Results from adult census income dataset

Maj. = 24,720; Min. = 7841	Border M5 = 5640–71.9% Border M8 = 5499–70.13%			I.R. = 31.7%
Oversampling technique	B%	Gen	Recall	*F*-measure
1.0: Original	0	0	0.625	0.684
2.1: SMOTE 8	0	7841	0.81	0.838
3.1: B-SMOTE *P′* & *P*: 8, 8	100	5499	0.772	0.803
4.1: AB-SMOTE *P′* & *P′*: 8, 8	100	5499	0.788	0.81
5.6: CAB-SMOTE SI: 5, 5, 7	140	7926	0.81	0.842
6.0: CAB-SMOTE SII: 5, 5, 2	100	5640	0.774	0.82
7.2: CAB-SMOTE SIII: 5, 5, 3	100	5640	0.779	0.821
8.2: HCAB-SMOTE: 5, 5, 3	86	4893	**0.827**	**0.852**

4.1 Results and Discussion

Tables 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16 show the results obtained from classifying the datasets before and after applying 100% oversampling. The tables display the number of majority instances (MAJ.), number of minority instances (Min.), number of instances within the borderline area at Mnn = 5 and at Mnn = 8 and their percentages from the minority samples which are used with B-SMOTE and the techniques that follows, the imbalanced ratio (I.R.) that is computed by dividing Min. over Maj., The oversampling technique number (x.0) with the value of x between 1 and 8 where it represent no oversampling, SMOTE, B-SMOTE, AB-SMOTE, CAB-SMOTE SI, SII, SIII and HCAB-SMOTE respectively, the version number that got the best F-measure result (0.y) with values between 0 and 7, associated with the technique name followed by k nearest neighbors, M nearest neighbors and k-means values used while applying these techniques under the oversampling technique column, a B% column that represent the percentage of generated instances over the borderline size to display how much the algorithm generated less instances than the former ones, Gen. column that recorded the number of generated new instances, Recall and F-measure columns that display their values, and highlighted the highest Recall and F-measure values.

After applying all the techniques using the same oversampling rate, HCAB-SMOTE and other techniques are compared by the difference in generated instances with Eq. (5) and F-measure difference with Eq. (6), and then results for each dataset are analyzed.

$$\text{Dif. GEN} = |\,(\text{GEN.HCAB} - \text{SMOTE}/\text{GEN.SMOTE}) - 1| \tag{5}$$

$$\text{Dif.}F\text{ - measure} = |\,(\text{F.HCABSMOTE}/\text{F.SMOTE}) - 1\,| \tag{6}$$

Table 7 displays the results obtained from the Abalone dataset, where HCAB-SMOTE generated 7.14% less instances and attained highest F-measure with 43.8% more than that of SMOTE, and generated same number of instances as B-SMOTE and AB-SMOTE but improved 40.9% and 59% in the F-measure values, respectively, followed by CAB-SMOTE SII and SIII that delivered same results because all borderline clusters were larger than half of the largest cluster.

Table 8 shows results obtained from the Customer Churn dataset. The results obtained without any oversampling had the highest recall and F-measure compared to the results achieved after applying SMOTE, B-SMOTE, and AB-SMOTE, whereas after applying the CAB-SMOTE the results outperformed the former ones. The highest F-measure values was achieved by HCAB-SMOTE that generated 38.8% less than SMOTE and 14% increase in the F-measure value and generated 19% less than CAB-SMOTE SIII gaining 1.3% more in F-measure value. CAB-SMOTE SIII followed it for the second highest F-measure values for this dataset outperforming SII because it oversampled all clusters.

Table 9 displays the results obtained from oversampling the Herberman dataset. The HCAB-SMOTE algorithm outperformed other techniques and achieved the highest F-measure values. It generated 37% fewer instances than SMOTE and 18.35% increase in F-measure value. In comparison with CAB-SMOTE SII, they both generated same number of instances, but HCAB-SMOTE delivered 16% more F-measure value which translated to the undersampling gain. CAB-SMOTE SIII achieved the second highest recall and F-measure values.

Table 10 displays the results obtained from oversampling the employee attrition dataset. The highest value of recall and F-measure was obtained after applying HCAB-SMOTE which oversampled 42.79% less than SMOTE and attained 17.96% more classification accuracy. CAB-SMOTE SII & SIII achieved second highest recall and F-measure values where they generated 74.8% more and achieved 3.3% less in F-measure value.

Tables 11 display the classification accuracy results from using Telco Customer Churn dataset. HCAB-SMOTE outperformed the other oversampling methods by giving the highest F-measure value with 49.5% less generated instances than of SMOTE and 10.3% more in the classification accuracy. SMOTE achieved the second highest F-measures values. CAB-SMOTE SI reached highest recall value even while oversampling less than SMOTE, but it had lower F-measure meaning that it was misclassifying an actual negative instance as a positive one more than SMOTE, which would be costly misclassification such as giving more advertisements to a loyal client which was not going to churn. HCAB-SMOTE generated same number of instances as CAB-SMOTE SII but achieved 12.5% more F-measure value due to the undersampling process.

Table 12 displays the classification accuracy results from Flare dataset where HCAB-SMOTE got highest results followed by CAB-SMOTE SIII. HCAB-SMOTE generated 9.3% less instances than SMOTE and delivered 50.8% increase in classification accuracy and generated the same number of instances as CAB-SMOTE SII but 15% in F-measure value due to the undersampling process.

Table 13 shows results from using Wine Quality dataset where HCAB-SMOTE outperformed the other techniques followed by CAB-SMOTE SII. HCAB-SMOTE managed to generate 33.9% less than SMOTE and delivering 199% more F-measure value, and same number of generated instances of CAB-SMOTE SII but with 40% increase in F-measure values.

Table 14 displays results from Sales Win-Loss dataset where HCAB-SMOTE get highest values followed by CAB-SMOTE SII. HCAB-SMOTE oversampled 13.9% less than SMOTE and 3.1% more *F*-measure and generated the same number of instances as CAB-SMOTE SII but with 2.7% increase in *F*-measure value.

Table 15 displays classification accuracy from Bank Subscription dataset, where HCAB-SMOTE get higher results followed by CAB-SMOTE SII & III. HCAB-SMOTE generated 18.1% less than SMOTE and 11.39% more *F*-measure, and it generates the same number of instances as CAB-SMOTE SII and SIII but delivered 8.5 increase in *F*-measure value.

Table 16 displays results achieved from Adult dataset, where HCAB-SMOTE got the highest accuracy results while generating 38.2% less than SMOTE and 1.67% increase in *F*-measure values and generating 13.2% less than CAB-SMOTE SII and delivering 3.9% *F*-measure increase. CAB-SMOTE SI oversampled more than SMOTE because the borderline was clustered into 7 clusters and divided into small clusters and a large one so it oversampled the small cluster to reach the largest one, leading to this high number of generated instances leading to higher *F*-measure value.

Table 17 displays results from oversampling Amazon Employees access dataset where HCAB-SMOTE generated 8% less instances than SMOTE and gained 17.9% more *F*-measure value. HCAB-SMOTE outperformed the other techniques, followed by CAB-SMOTE SI because it oversampled more instances than the original SMOTE.

Table 18 displays the results from using Click Prediction dataset where HCAB-SMOTE generated 14% less than SMOTE and obtained 17% increase in *F*-measure value. CAB-SMOT SI got the highest *F*-measure because it generated 16% more than SMOTE and definitely more than other techniques which made it logical to get the highest *F*-measure value.

Table 17 Results from Amazon employee access dataset

Maj. = 30,872; Min. = 1897	Border M5 = 1744–91.9% Border M8 = 1752–92.3%			I.R. = 6.14%
Oversampling technique	*B*%	Gen	Recall	*F*-measure
1.0: Original	0	0	0.17	0.265
2.1: SMOTE $K=8$	0	1897	0.523	0.617
3.0: B-SMOTE P' & P: 5, 5	100	1744	0.486	0.577
4.1: AB-SMOTE P' & P': 8, 8	100	1752	0.488	0.587
5.7: CAB-SMOTE SI: 8, 8, 7	110	1937	0.602	0.71
6.1: CAB-SMOTE SII: 8, 8, 2	100	1752	0.587	0.704
7.1: CAB-SMOTE SIII: 8, 8, 2	100	1752	0.587	0.704
8.2: HCAB-SMOTE: 5, 5, 3	100	1744	**0.621**	**0.728**

Table 18 Results from click prediction dataset

Maj. = 33,220; Min. = 6728	Border M5 = 6300–93.6% Border M8 = 6132–91.1%			I.R. = 20.2%
Oversampling technique	*B*%	Gen	Recall	*F*-measure
1.0: Original	0	123,456	0.041	0.077
2.1: SMOTE $K=8$	0	6728	0.419	0.575
3.3: B-SMOTE P' &N: 5, 5	100	6132	0.406	0.565
4.0: AB-SMOTE P' & P': 5, 5	100	6300	0.397	0.555
5.7: CAB-SMOTE SI: 8, 8, 7	127	7812	**0.563**	**0.705**
6.0: CAB-SMOTE SII: 5, 5, 2	100	6300	0.507	0.667
7.0: CAB-SMOTE SIII: 5, 5, 2	100	6300	0.507	0.667
8.4: HCAB-SMOTE: 8, 8, 3	94	5765	0.532	0.676

After observing the results, the following points were noticed:

- CAB-SMOTE SI oversamples all the borderline clusters to reach the size of the largest one, without oversampling within the largest cluster. Therefore, it might oversample less or more than B-SMOTE and AB-SMOTE. Its classification results outperform those techniques even while generating less instances, which indicate that oversampling within a cluster is more efficient than oversampling within the whole danger region.
- CAB-SMOTE SII oversamples all the clusters which are larger than half of the largest cluster, and achieves better classification accuracy than CAB-SMOTE SI when generating slightly fewer number of instances and definitely achieving better results while generating same number of instances, indicating that the oversampling large clusters are more important than small clusters within the borderline region.
- CAB-SMOTE SIII oversamples all the borderline clusters, when all the clusters are larger than half of the largest cluster CAB-SMOTE SIII obtain same classification

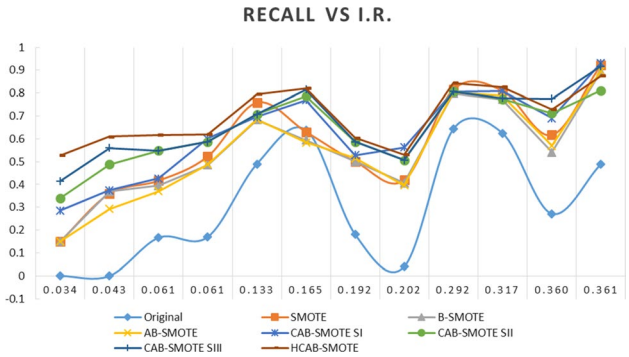

Fig. 13 Recall $_{\text{V-AXIS}}$ against the datasets Imbalanced ratios $_{\text{H-AXIS}}$

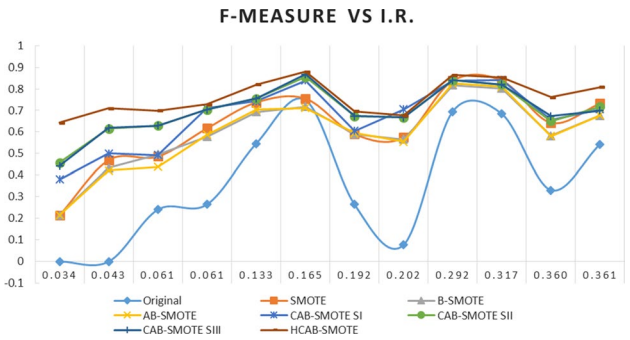

Fig. 14 *F*-measure $_{\text{V-AXIS}}$ against the datasets Imbalanced ratios $_{\text{H-AXIS}}$

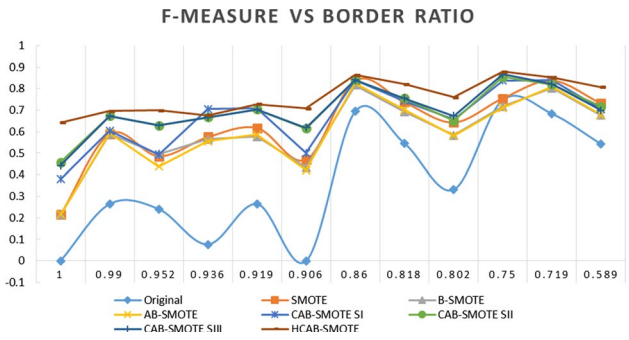

Fig. 16 *F*-measure $_{\text{V-axis}}$ against Border Ratio $_{\text{H-axis}}$

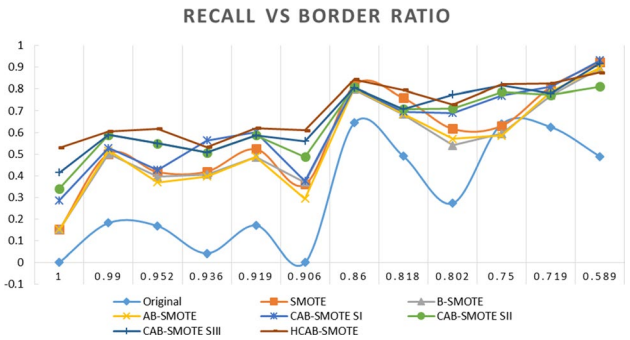

Fig. 15 Recall $_{\text{V-axis}}$ against Border ratio $_{\text{H-axis}}$ computed in the dataset

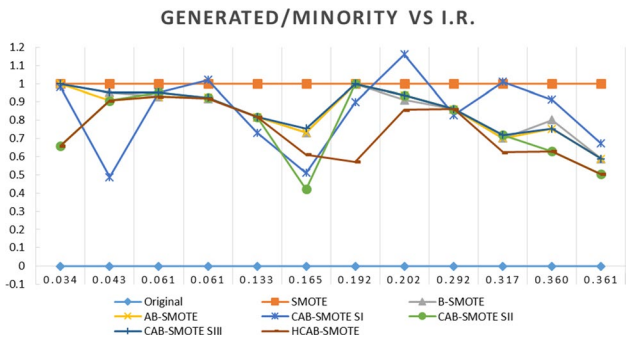

Fig. 17 Generated to minority instances ratio $_{\text{V-axis}}$ versus I.R. ratio $_{\text{H-axis}}$

accuracy as of CAB-SMOTE SII generating the same number of instance, but when some clusters are smaller than half of the largest one CAB-SMOTE SIII generates more instances and obtain better results. Therefore, SII delivers good results without oversampling clusters smaller than half of the largest one, which leads us to use SII within HCAB-SMOTE to minimize generating instances.

– HCAB-SMOTE outperforms the other techniques due to the combination of under- and oversampling processes which remove small clusters of minority borderline instances and negative borderline instances, improving the classification results from two sides at the same time.

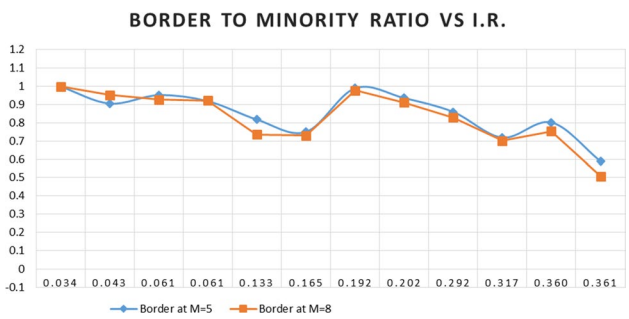

Fig. 18 Border ratio $_{\text{V-axis}}$ to I.R. $_{\text{H-axis}}$ in the dataset

Figures 13 and 14 display the recall and *F*-measure values against the percentage of imbalanced ratio found in the tested datasets. Imbalanced ratios are calculated by dividing the number of minority instances over the number of majority instances. The graph display different I.R. values of each original dataset within the experiment on the horizontal-axis and then display on the vertical axis the Recall and *F*-measure values after applying the decision tree algorithm on the original data and after oversampling with different techniques. The most problematic one is 0.034, and the best one is 0.361. It shows that when the I.R. is small, the recall and *F*-measure values are also small and they increase after applying oversampling techniques. HCAB-SMOTE outperforms other oversampling techniques through most of the datasets.

Figures 15 and 16 display the Recall and *F*-measure values against the borderline percentage from the minority class. When the border ratio is equal to one, every minority class is considered as a border. When the border-to-minority ratio is more than 90%, the original data get very low recall and *F*-measure values, which is because more than 90% of the minority instances have more than 3 negative

instances within their 5 nearest neighbors thus having too many noise where the classifier could not build up good model to predict minority instances. This is improved by applying oversampling techniques.

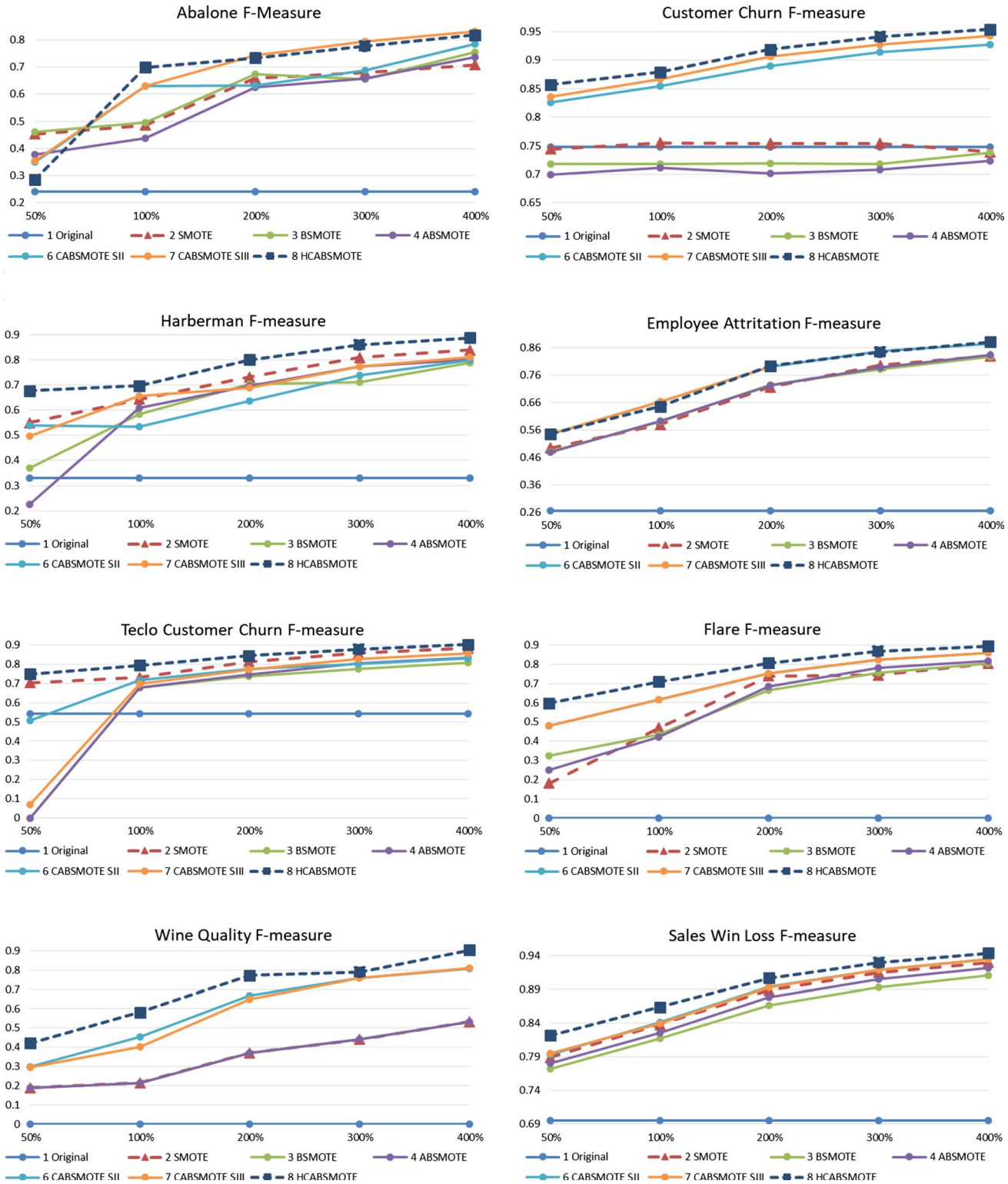

Fig. 19 *F*-measure values of increasing oversampling percentages over all datasets

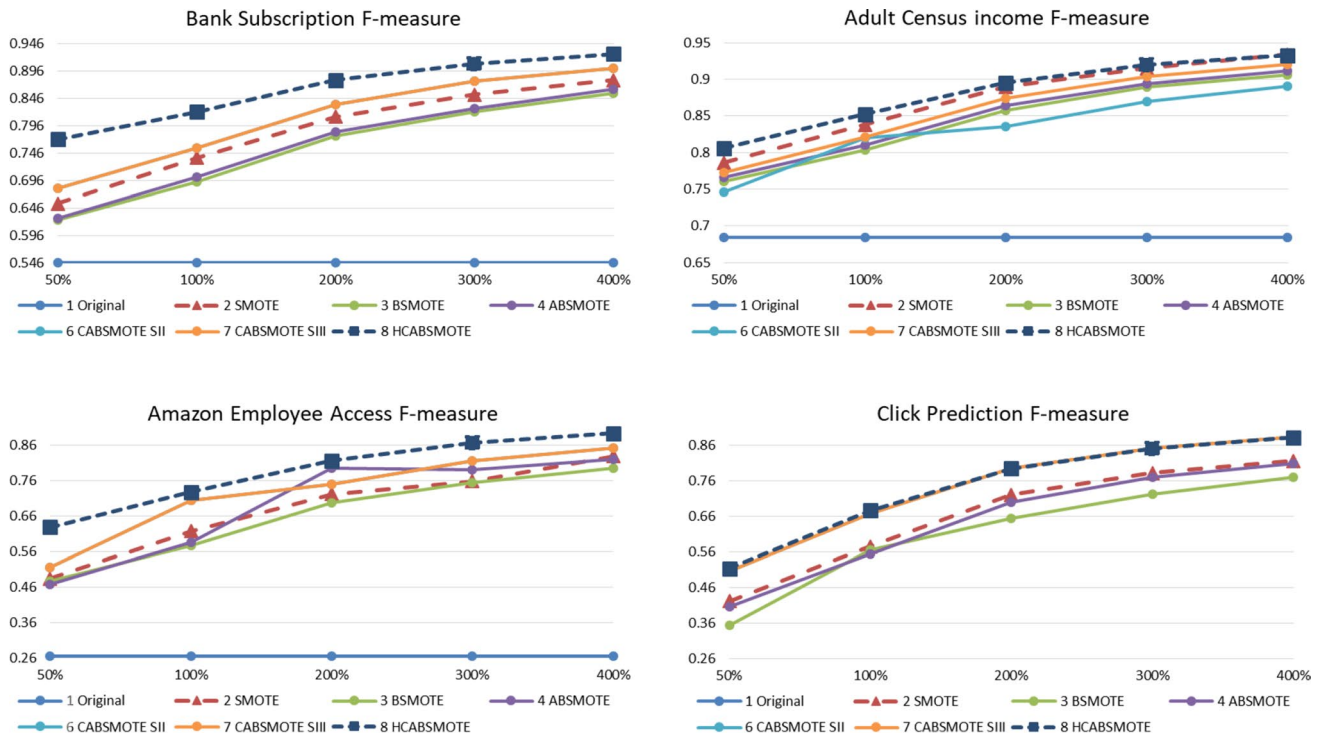

Fig. 19 (continued)

Figure 17 displays the graph of each datasets I.R. and the newly generated over minority instances ratio. SMOTE has a value of 1, because it doubles the number of minority instances, while CAB-SMOTE SI sometimes oversamples more than the number of minority instances where it achieved a value of 1.16 due to the differences between clusters sizes. There is no clear trend in within this graph, but it is noticeable that HCAB-SMOTE oversamples less than other techniques.

Figure 18 shows the relation between the ratios of border to minority against the I.R. of each dataset. It indicates that datasets chosen in the experiment that have low I.R. tend to have border that covers more than 90% of the minority class, and when the I.R. increases, the border shrinks, thus enabling the border-related techniques to generate fewer instances.

Figure 19 displays the *F*-measure values obtained while applying different oversampling methods and rates starting from 50% up to 400% oversampling. The approach number 5 CAB-SMOTE SI was not included with this experiment because it oversamples the instances inside the clusters to reach the same number of instances found in the largest cluster, and thus the number of oversampling is limited to the number of instances found in the largest cluster. These figures give a clear indication that all the oversampling techniques provide better classification results when increasing the oversampling rate. CAB-SMOTE and HCAB-SMOTE managed to provide higher classification accuracy results than the original oversampling techniques, verifying that oversampling special areas within the boundaries of the borderline is better than increasing the density inside all the border regions as a whole or around its boundaries. Also each technique sustained its ranking within other oversampling throughout increasing the oversampling rate.

5 Conclusion

This article proposed AB-SMOTE that focuses on limiting the oversampling region to be within the borderline area. Then the article developed CAB-SMOTE to cluster the border region and oversample each cluster independently. Finally, HCAB-SMOTE was proposed to minimize the number of generated instances while increasing the classification accuracy by creating new instances based on border region cluster sizes using CAB-SMOTE SII, and applying undersampling to remove noisy minority and majority instances from the border.

Classification accuracy is evaluated by measuring the accuracy of classifying the minority instances to be in their actual class, using recall and *F*-measure values. On the one hand, experimental results show that CAB-SMOTE SII can perform efficiently if the user does not want to under sample the data, while on the other hand if the user does not mind using undersampling and uses HCAB-SMOTE, it will boost the classification accuracy for the minority

class with same or fewer generated instances than CAB-SMOTE SII.

Future work will focus on using density-based approaches for clustering the borderlines rather than using classical KNN, because datasets with no clear borders need special analysis for finding the proper clusters.

References

1. Sun, A.; Lim, E.P.; Liu, Y.: On strategies for imbalanced text classification using SVM: a comparative study. Decis. Support Syst. **48**(1), 191–201 (2009)
2. Tek, F.B.; Dempster, A.G.; Kale, I.: Parasite detection and identification for automated thin blood film malaria diagnosis. Comput. Vis. Image Underst. **114**(1), 21–32 (2010)
3. Qureshi, S.A.; Rehman, A.S.; Qamar, A.M.; Kamal, A.; Rehman, A.: Telecommunication subscribers' churn prediction model using machine learning. In: Eighth International Conference Digital Information Management (ICDIM 2013), September, pp. 131–136 (2013)
4. "Keel Datasets, Wine Quality." https://sci2s.ugr.es/keel/dataset.php?cod=1322. Accessed 21 Aug 2019
5. Bekkar, M.; Alitouche, D.; Akrouf, T.; AkroufAlitouche, T.: Imbalanced data learning approaches review. Data Min. Knowl. **3**(4), 15–33 (2013)
6. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
7. Bunkhumpornpat, C.; Sinapiromsaran, K.; Lursinsap, C.: Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 5476 LNAI, pp. 475–482 (2009)
8. Chawla, N.V.; Lazarevic, A.; Hall, L.O.; Bowyer, K.W.: SMOTE-Boost: improving prediction of the minority class in boosting, pp. 107–119 (2003)
9. Han, H.; Wang, W.; Mao, B.: Borderline-SMOTE : a new over-sampling method in imbalanced data sets learning, pp. 878–887 (2005)
10. Bach, M.; Werner, A.; Żywiec, J.; Pluskiewicz, W.: The study of under- and over-sampling methods' utility in analysis of highly imbalanced data on osteoporosis. Inf. Sci. **384**, 174–190 (2017)
11. Douzas, G.; Bacao, F.; Last, F.: Improving imbalanced learning through a heuristic oversampling method based on *k*-means and SMOTE. Inf. Sci. **465**, 1–20 (2018)
12. Elhassan, A.T.; Aljourf, M.: Classification of imbalance data using Tomek Link (T-Link) combined with random under-sampling (RUS) as a data reduction method. J. Inform. Data Min. **1**(2), 1–12 (2016)
13. Oskouei, R.J.; Bigham, B.S.: Over-sampling via under-sampling in strongly imbalanced data. Int. J. Adv. Intell. Paradig. **9**(1), 58 (2017)
14. Japkowicz, N.: Learning from imbalanced data sets: a comparison of various strategies. In: AAAI Workshop Learning from Imbalanced Data Sets, vol. 68, pp. 10–15 (2000)
15. Stefanowski, J.; Wilk, S.: Selective pre-processing of imbalanced data for improving classification performance. In: Data Warehousing and Knowledge Discovery (Lecture Notes Computer Science Series 5182), pp. 283–292 (2008)
16. Laurikkala, J.: Improving identification of difficult small classes by balancing class distribution. Inf. Sci. (2001)
17. "Weka." https://www.cs.waikato.ac.nz/ml/weka/index.html. Accessed 7 Jan 2020
18. Fernández, A.; López, V.; Galar, M.; Del Jesus, M.J.; Herrera, F.: Analysing the classification of imbalanced data-sets with multiple classes: binarization techniques and ad-hoc approaches. Knowl. Based Syst. **42**, 97–110 (2013)
19. Keel Datasets, Abalone9-18. http://sci2s.ugr.es/keel/dataset.php?cod=116. Accessed 21 Aug 2019
20. Crowd Analytix. http://www.crowdanalytix.com/contests/why-customer-churn/. Accessed:21 Aug 2019
21. IBM Analytics Telco Customer Churn Dataset. https://www.kaggle.com/blastchar/telco-customer-churn#WA_Fn-UseC_-Telco-Customer-Churn.csv. Accessed 21 Aug 2019
22. Dua, C.; Dheeru; Graff: UCI Machine Learning Repository (2017). http://archive.ics.uci.edu/ml. Accessed 21 Aug 2019
23. Haberman, S.J.: Generalized residuals for log-linear models. In: Proceedings of 9th International Conference on Biometrics, pp. 104–122 (1976)
24. IBM & Kaggle Employee Attrition. https://www.kaggle.com/patelprashant/employee-attrition#WA_Fn-UseC_-HR-Employee-Attrition.csv. Accessed 21 Aug 2019
25. Keel Datasets, Solar Flare. https://sci2s.ugr.es/keel/dataset.php?cod=1331#sub1. Accessed 21 Aug 2019
26. IBM Analytic, Win Loss. https://github.com/vkrit/data-science-class/blob/master/WA_Fn-UseC_-Sales-Win-Loss.csv. Accessed 21 Aug 2019
27. Moro, S.; Laureano, R.M.S.; Cortez, P.: Using data mining for bank direct marketing: An application of the CRISP-DM methodology. In: ESM 2011–2011 European Simulation and Modelling Conference 2011, no. Figure 1, pp. 117–121 (2011)
28. Kohavi, R.; Becker, B.: Adult Census Income (1996). http://archive.ics.uci.edu/ml/datasets/Adult. Accessed 7 Jan 2020
29. K. A. E. A. Challenge: No Title. https://www.kaggle.com/c/amazon-employee-access-challenge. Accessed 7 Jan 2020
30. Cup, K.: No Title (2012). https://www.openml.org/d/1220. Accessed 7 Jan 2020
31. Cervantes, J.; Garcia-Lamont, F.; Rodriguez, L.; López, A.; Castilla, J.R.; Trueba, A.: PSO-based method for SVM classification on skewed data sets. Neurocomputing **228**, 187–197 (2017)
32. López, V.; Fernández, A.; Herrera, F.: On the importance of the validation technique for classification with imbalanced datasets: addressing covariate shift when data is skewed. Inf. Sci. **257**, 1–13 (2014)
33. Saito, T.; Rehmsmeier, M.: The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLoS ONE **10**(3), 1–21 (2015)