IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 11, 2020, accepted March 19, 2020, date of publication March 24, 2020, date of current version April 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2983003

SMOTEFUNA: Synthetic Minority Over-Sampling Technique Based on Furthest Neighbour Algorithm

AHMAD S. TARAWNEH[1], AHMAD B. A. HASSANAT[2,3,4], (Member, IEEE), KHALID ALMOHAMMADI[3], DMITRY CHETVERIKOV[1,5], AND COLIN BELLINGER[6]

[1]Department of Algorithms and Their Applications, Eötvös Loránd University, 1117 Budapest, Hungary
[2]Department of Computer Science, Mutah University, Karak 61711, Jordan
[3]Department of Computer Science, Community College, University of Tabuk, Tabuk 71491, Saudi Arabia
[4]Industrial Innovation and Robotics Center, University of Tabuk, Tabuk 71491, Saudi Arabia
[5]Institute for Computer Science and Control, Budapest, Hungary
[6]National Research Council, Ottawa, ON K1A 0R6, Canada

Corresponding author: Ahmad S. Tarawneh (ahmad.trwh@gmail.com)

The work of Ahmad S. Tarawneh was supported in part by the Hungarian Government, and in part by the European Social Fund, under Project EFOP-3.6.3-VEKOP-16-2017-00001 (talent management in autonomous vehicle control technologies). The work of Dmitry Chetverikov was supported by a project ED_18-1-2019-0030 (application domain specific highly reliable IT solutions subprogramme) implemented with the support of the National Research, Development, and Innovation Fund of Hungary, financed under the Thematic Excellence Programme Funding Scheme.

ABSTRACT Class imbalance occurs in classification problems in which the ''normal'' cases, or instances, significantly outnumber the ''abnormal'' instances. Training a standard classifier on imbalanced data leads to predictive biases which cause poor performance on the class(es) with lower prior probabilities. The less frequent classes are often critically important events, such as system failure or the occurrence of a rare disease. As a result, the class imbalance problem has been considered to be of great importance for many years. In this paper, we propose a novel algorithm that utilizes the furthest neighbor of a candidate example to generate new synthetic samples. A key advantage of SOMTEFUNA over existing methods is that it does not have parameters to tune (such as K in SMOTE). Thus, it is significantly easier to utilize in real-world applications. We evaluate the benefit of resampling with SOMTEFUNA against state-of-the-art methods including SMOTE, ADASYN and SWIM using *Naive Bayes* and *Support Vector Machine* classifiers. Also, we provide a statistical analysis based on Wilcoxon Signed-rank test to validate the significance of the SMOTEFUNA results. The results indicate that the proposed method is an efficient alternative to the current methods. Specifically, SOMTEFUNA achieves better 5-fold cross validated ROC and precision-recall space performance.

INDEX TERMS Binary classification, data mining algorithm, furthest neighbor, imbalance problem, SMOTE.

I. INTRODUCTION

Class imbalance appears in classification data where the training samples from one class (majority class) significantly outnumber the samples from the other class (minority class) [1]. The fundamental issue in class imbalance is that classifiers induced on imbalanced training sets exhibit a predictive bias associated with weak performance on the minority class.

The predictive bias is especially problematic because the minority class is often of critical importance [2]. This is indeed the case in real-world problems such as disease detection [3], credit card fraud detection [4], gene profiling [5], invoice classification [6], face image retrieval [7], content-based image retrieval [8], authentication [9], radiation detection [10], *etc*.

The associate editor coordinating the review of this manuscript and approving it for publication was Gianluigi Ciocca.

Formally, given a supervised machine learning dataset D, with n examples, which belong to m classes $\{C_1, C_2, C_3, \ldots, C_m\}$, if any $| C_i | \ll | C_j |$ then D is called an imbalanced dataset. There are several approaches used to solve this problem such as

- Collecting more examples belonging to the minority class(es).
- Adjusting the loss function to assign a higher cost to the failed detection of the minority class instances, or a lower cost to the majority class [11].
- Re-sampling the training set, *i.e.,* oversampling or synthetically sampling the minority class, undersampling the majority class, or applying a combination of these methods.

In practice, the collection of more data is limited by the associated time and cost constraints. Moreover, in some domains, the minority class is truly rare, and thus, a sufficient number of samples cannot be collected at any cost. Cost adjusting has been shown to be an effective approach to deal with imbalance; however, it generally requires adjustments to the classification algorithm and/or the prior knowledge of appropriate misclassification costs. Alternatively, resampling is an easy to apply domain independent option, which has been shown to be equivalent to cost adjustment under certain circumstances [12]. Within the realm of resampling methods, synthetic oversampling is particularly effective because it avoids the loss of information that is common in random undersampling and overfitting which can occur due to random oversampling [13].

In this paper, we propose a novel method of synthetic oversampling the minority class, SOMTEFUNA. Unlike Synthetic Minority Oversampling TEchnique (SMOTE) [13], which generates samples within the convex-hull formed by the minority samples, our approach is based on finding a *potential space* in the feature space, which forms a (hyper) cuboidal shape between selected minority examples. This potential space is found by connecting the furthest neighbours in the minority class. The key intuition is that within the potential space, it is appropriate to generate minority samples.

Forming the potential space from the furthest neighbour has the fundamental advantage of improving diversity in the synthetic set, and thus, the generality of the induced classifier. Alternatively, the traditional methods that generate samples between k-nearest minority class examples have a tendency to produce synthetic samples that are near replicas of their seeds. This risks the induction of model that overfits the minority class. Thus, we postulate that SOMTEFUNA improves the interpolation between the minority examples and helps to avoid overfitting.

With SOMTEFUNA, candidate examples are generated at random points within the potential space. In the final step, a generated candidate is saved as a synthetic minority sample if and only if its nearest neighbour in the training set is also a minority class example; otherwise it is deleted. This ensures that the synthetic samples do not encroach deep into the majority class space, a problem which can occur with other methods.

For our analysis, we undertake 5-fold cross validated experiments on 33 benchmark datasets with *Naive Bayes* and *Support Vector machines* (SVM) classifiers. We utilize ROC and precision-recall curves for evaluation and empirically show that SOMTEFUNA is an efficient alternative to current state-of-the-art synthetic oversampling algorithms including SMOTE, ADASYN and SWIM.

The rest of this paper is structured as follows: Section 2 highlights the relevant recent work on class imbalance. In Section 3, we present the proposed SOMTEFUNA oversampling algorithm, and in Section 4, we present and discuss the experimental results.

II. RELATED WORK

SMOTE [14] is one of the most widely applied methods used to mitigate the negative effects of class imbalance. It interpolates synthetic instances between nearest neighbours in the set of minority class instances in the training set. Thus, a synthetic sample is formed as a combination of the features of seed examples and randomly selected k-nearest neighbours. The k parameter must be set by the user. The initial version of the SMOTE algorithm only applied synthetic oversampling. However, a combination of synthetic oversampling and undersampling can be effective [15]. The original study empirically evaluated SMOTE on 9 benchmark datasets and found it to improve classifier performance.

SMOTEBoost [16] is another approach for learning from imbalanced datasets; this method combines SMOTE and a standard boosting algorithm. It generates new examples from the minority class by applying SMOTE in each round of boosting. This enables each learner to learn more from the minority class examples. SMOTEBoost was applied to four imbalanced datasets, and the results showed a slight F-value improvement in the prediction compared to the original SMOTE.

Pan and co-workers [17] proposed two sampling methods to improve imbalanced datasets. The first is Adaptive-SMOTE, which is intended to improve SMOTE by selecting collections of *inner* and *danger* data from the minority class. This is applied to firm up the distributional characteristics of the original data. The second proposed method is based on Gaussian oversampling, which combines dimension reduction with the Gaussian distribution. Both of these methods were applied on 15 imbalanced datasets. The empirical evaluation showed good results obtained by both methods. The performance of non-minority classes, however, was found to degrade after sampling.

Another interesting method for learning from imbalanced datasets is the adaptive synthetic sampling (ADASYN) [18]. This method uses a weighted distribution over examples that belong to the minority class. This is used to assess and address the learning difficulty. Specifically, more new examples are created for hard-to-learn examples, compared to those easier-to-learn minority examples. This approach

improves the learning process by reducing the bias introduced by the imbalanced data and shifting the classification decision boundary towards the hard-to-learn examples. This method was applied to five imbalanced machine learning datasets. The comparison results showed that the F-values of the ADASYN were slightly better than those of SMOTE in three datasets.

Sampling with the majority class (SWIM) [19] is a framework proposed for synthetic oversampling in extreme imbalance classification. Opposite to most of the aforementioned methods, the key feature of this method is the exploitation of the density of the well-represented majority class. This is used to guide the generation of new examples of the minority class. This method was evaluated on 25 datasets with extreme levels of imbalance. The results show improvement in the classification performance compared to some of the existing oversampling techniques. The greatest improvement was shown to occur on datasets with the most extreme level of imbalance.

Siriseriwan and Sinapiromsaran proposed a parameter-free adaptive algorithm called Adaptive Neighbor Synthetic Minority Oversampling Technique (ANSMOT) to change the number of neighbors required for oversampling across different minority regions [20]. The major advantage of ANSMOT compared to SMOTE is that it removes the need for the dataset parameter K. Their experiments on 14 imbalanced datasets showed that the F-measures have improved when using ANSMOT comparing to the other methods tested.

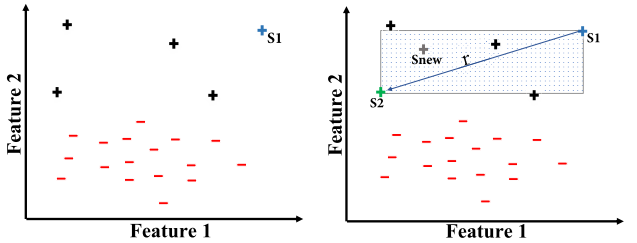
An informative review of oversampling methods can be found in [21], where a comprehensive empirical analysis of 85 minority oversampling methods is presented, discussed, and evaluated on 104 imbalanced datasets.

There are other methods proposed in the literature to enhance learning from imbalanced datasets, these include but not limited to [22]–[30]. Despite the success of these methods in real-world applications, all of them have their own pros and cons, and none of them is perfect, and therefore, there is room for enhancement in this research line. This paper proposes an efficient method to enhance learning from imbalanced datasets, whose results indicate that it is better than some popular, state-of-art methods.

III. METHODOLOGY

The proposed method is prefaced on generating synthetic minority samples at random points within a potential space (hyper cuboidal) formed from the furthest neighbours from randomly selected minority class examples (seed instances).

Figures 1(a) and 1(b) show hypothetical minority and majority examples (left) and a candidate synthetic instance generated from the minority seed and its furthest neighbour (right). As can be seen in Figure 1(b), one example $\vec{S_1}$ is randomly selected from the minority class (shown as a blue cross). The Furthest Neighbour (FN) algorithm finds the example with the max distance r from the selected seed example. In the figure, we denote it as $\vec{S_2}$ (shown in Figure 1(b)). The new synthetic sample is produced



(a) hypothetical examples (b) Candidate location (potential space)

FIGURE 1. Two hypothetical classes (a) and the potential space for the new examples (b).

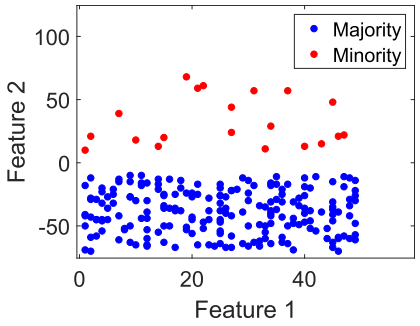
by taking a random value between each pair of the two vectors. For instance, assuming that $\vec{S_1} = \{X_1, Y_1\}$ and $\vec{S_2} = \{X_2, Y_2\}$, the synthetic sample will be $\vec{S}_{new} = \{\lambda, \gamma\}$, where $\lambda = Random\left(min\left(X_1, X_2\right), max\left(X_1, X_2\right)\right)$ and $\gamma = Random\left(min\left(Y_1, Y_2\right), max\left(Y_1, Y_2\right)\right)$. This emphasizes the distinction that SMOTE would generate a new sample at a random point on the line connecting $\vec{S_1}$ and $\vec{S_2}$, whereas SMOTEFUNA generates the sample between each feature value of the two examples, i.e., each feature of the new sample is located within the ranges of the two examples chosen from the dataset. In this case, the produced example is forced to be located within the cuboidal potential space formed between the two real examples from the dataset. Figure 1(b) presents the potential space as a shaded rectangle, and the hypothetical synthetic example $\vec{S}_{new}$ is a gray cross within the potential space (Figure 1(b)).

Selecting the furthest example provides an advantage of allowing for a more diverse set of synthetic samples. At the same time, the synthetic distribution is likely to be similar to the true minority distribution because the synthetic examples are generated from real minority instances. This is because the nearest neighbour constraint on each synthetic sample ensures that the sample does not intrude on the majority class.

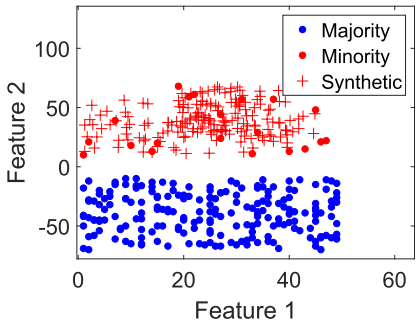
Figure 2 shows a scenario of linearly separable examples and their (oversampling) results using SMOTEFUNA compared to the result of SMOTE, while Figure 3 presents a variation on our algorithm using nearest neighbour examples in the augmentation process.

As can be clearly seen in Figures 2 and 3, the proposed method provides better coverage of the minority class. In contrast, using the nearest examples in the process of augmentation creates clusters of new examples in the original minority data. This can distort the resulting distribution when outlier examples are located in the decision boundary of the majority class (overlapping) [31], [32].

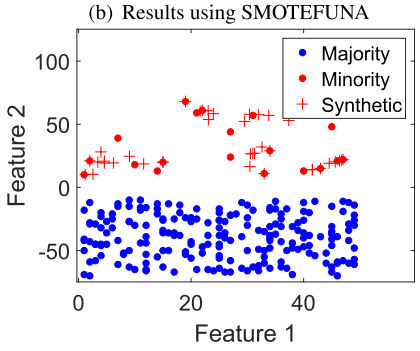
Data complexity has been shown to be a major factor contributing to the degradation of classifier performance in imbalanced domains [33]. A key aspect of this complexity is overlapping class distributions [34]–[37]. Figure 4(a) depicts an artificial classification setting in which the classes are overlapping. Post-hoc cleaning of the augmented datasets is a common way to deal with overlap after synthetic oversampling. Alternatively, SMOTEFUNA has a built-in check to



(a) Original data (scenario)

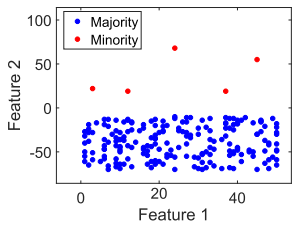


(b) Results using SMOTEFUNA

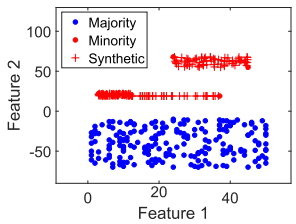


(c) Results using SMOTE

FIGURE 2. Results of SMOTEFUNA algorithm on a simple linearly separable space compared to SMOTE.



(a) Scenario of tow linearly separable classes

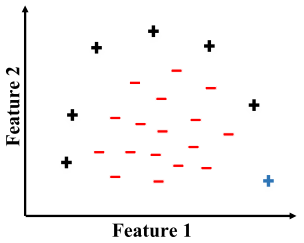


(b) Results when adding the examples based on nearest the neighbour.

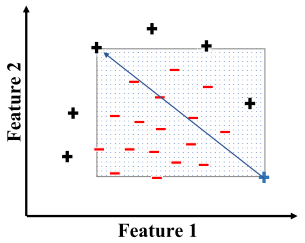
FIGURE 3. The affect of selecting the nearest neighbours for creating synthetic examples.

help prevent synthetic samples from being generated inside the majority class. For domains with overlapping classes, the potential space will overlap with the majority class. Thus, we do not want to blindly generate synthetic samples in the overlapping areas. This is demonstrated in Figure 4(b).

To prevent synthesizing instances in the majority class space, we add a constraint based on the candidate sample's nearest neighbour. Specifically, if its nearest neighbour is



(a) An overlapped distribution of the data examples



(b) False potential space

FIGURE 4. An example of false potential space for creating new examples.

Algorithm 1 SMOTEFUNA Algorithm

Input: Imbalanced dataset D
Output: Balanced dataset $\hat{D}$

$Minority \leftarrow D(Minoritysubset)$
$Majority \leftarrow D(Majoritysubset)$
$Rate \leftarrow length(Majority) - length(Minority)$
$i \leftarrow 0$
while $i \leq$ Rate **do**
  $n = Random(1, length(Minority)$
  $\vec{S}_1 \leftarrow Minority(n)$
  $\vec{S}_2 \leftarrow$ The vector with the maximum distance d from $\vec{S}_1$ in Minority
  $\vec{S}_{new} \leftarrow zeros(length(\vec{S}_1))$
  for $j = 1$ **to** $length(\vec{S}_1)$ **do**
    $S_{newj} = Random\left(min\left(S_{1j}, S_{2j}\right), max\left(S_{1j}, S_{2j}\right)\right)$
  if $\theta \leq \beta$ **then**
    $Synthetic(i) = \vec{S}_{new}$
    $i++$
  else
    go to 5
$\hat{D} = HConcatenation(Minority, Synthetic, Majority)$
$\hat{D} = RandomShuffle(\hat{D})$

from the majority class, then the candidate is discarded and new random candidate is generated; otherwise, the candidate is accepted and added to the synthetic set.

Algorithm 1 illustrates the steps employed by SMOTEFUNA to generate the synthetic minority set. The term $\theta = min(d(\vec{S}_{new}, MinorC))$ is the minimum distance between the generated example and its nearest example in the minority class, and $\beta = min(d(\vec{S}_{newj}, MajorC))$ is the minimum distance between the generated example and its nearest example in the majority class. $MinorC$ and $MajorC$ are the minority and majority examples closest to $\vec{S}_{new}$, respectively. The terms θ and β in Algorithm 1 are the core of the built-in checker, which validates any newly generated sample. According to our definition, if θ is smaller than or equal to β, then the generated sample is closer to the minority data, therefore, it can be created since, as we assume, it is more likely to be within the decision boundary of the minority class. Otherwise, if θ value is greater than β value, then the newly generated example is overlapping with the majority

data, thus it must be deleted. Step 6 selects a natural random number between 1 (assuming the index starts from 1) and the length of the minority set. *Minority* and *Majority* contain the minority and majority class samples from the main dataset D, respectively. *Minority(n)* returns the nth example in the minority set *Minority*. The distance function d can be calculated using any distance measurement. We utilized the Manhattan distance (Equation 1) because of its simplicity and computational efficiency:

$$d_k = \sum_{j=1}^{n} |S_1 j - S_k j|, \tag{1}$$

where d_k is the distance between the current (randomly selected) example $\vec{S_1}$ and the kth example in the dataset $\vec{S_k}$. The variable k varies from 1 to l, where $l = |Minority| - 1, n$ is the number of attributes (features) in the dataset. The distance d is calculated between $\vec{S_1}$ and all the other examples of the minority subset in order to find the furthest one (maximum distance) from $\vec{S_1}$. Step 9 initializes a vector of zeros whose length is the same as an actual example. *Synthetic(i)* is set to the ith instance of the synthetic set. *HConcatenation* preforms a horizontal concatenation of the three sets, *Minority*, *Synthetic* and *Majority*, in order to form the final balanced dataset. *RandomShuffle*($\hat{D}$) is an optional step to randomize the locations of the synthetic samples in the resulting training set. Applying the randomization may help with the induction of some classification methods. In particular, it prevents the learning from seeing all of the synthetic samples in succession during training, *e.g.,* in a single batch of neural net training.

Figure 5 shows scenarios of different data distributions and their corresponding oversampling results using SMOTEFUNA and SMOTE with $K = 5$.

Figure 5 provides a clear demonstration of the fact that SMOTE generates samples that both overlap with the majority class and result in less coverage of the minority class space than SMOTEFUNA. Alternatively, SMOTEFUNA produces examples within the same boundary of the minority with much better interpolation.

Figure 6 shows another scenario in which the data of the minority class are spread as separated groups around the data of the majority class.

As can be seen from Figure 6, even if the minority data is separated into different clusters, SMOTEFUNA generates the new samples within or near to each cluster. This ability of SMOTEFUNA to follow the distribution of the minority data is due to the aforementioned main factors, the shape of the potential space which provides the flexibility to generate new samples, and the built-in check, which approves the sample produced if it is near the minority class, and otherwise deletes it.

In Figure 7, we illustrate the invariance of SMOTEFUNA to outlier examples. By validating the generated example with its nearest neighbour, SMOTEFUNA makes the algorithm much less sensitive to a type of outlier where

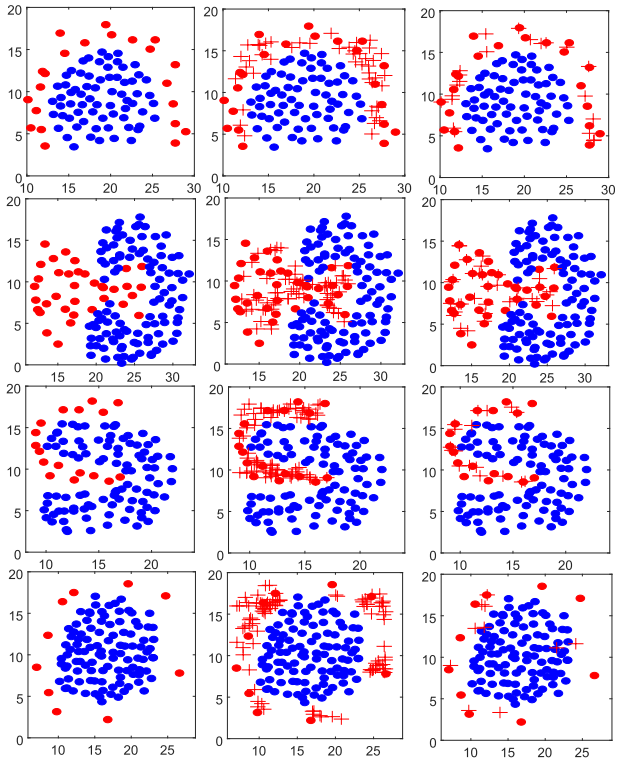


FIGURE 5. Different data distributions of minority and majority classes (left column), the results of oversampling using SMOTEFUNA (middle column) and SMOTE results (right column), blue points are the majority, red points are the minority and the red crosses are the new examples.

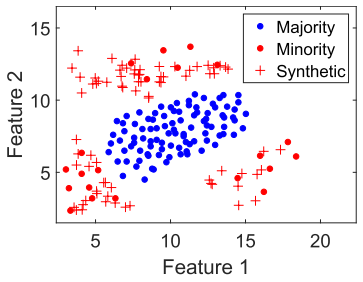
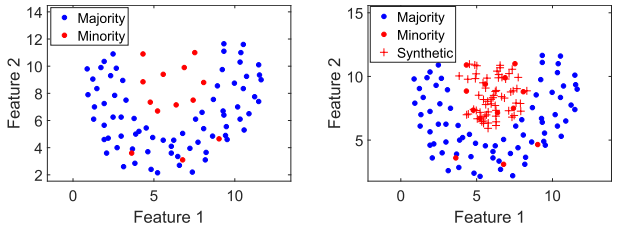


FIGURE 6. Minority data forms clusters around the majority data.

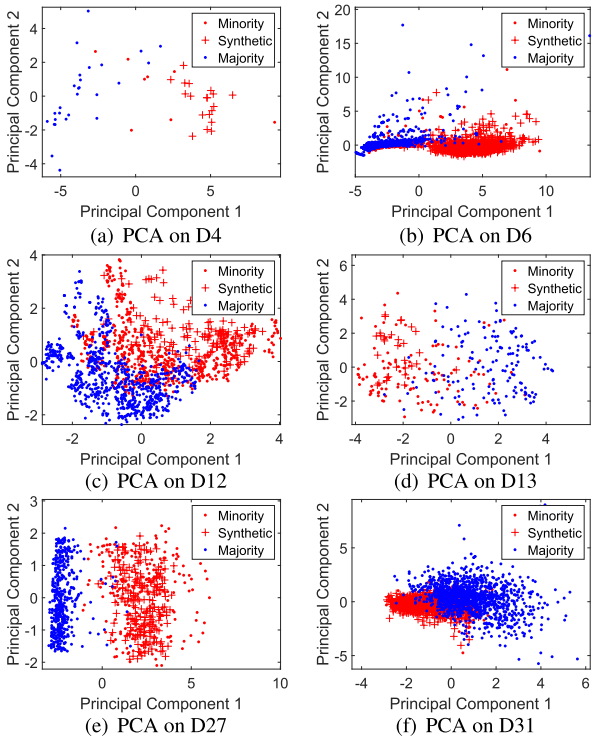
minority examples overlap with majority examples in the space. In particular, we present a scenario in which outlier instances from the minority class overlap with the majority class (Figure 7(a)), and illustrate that SMOTEFUNA does not reinforce these outliers by generating additional overlapping synthetic instances (Figure 7(b)). In SMOTE, users are required to apply additional post-hoc cleaning methods, such as the removal of Tomek links [38], to prevent this.

Figure 8 visualizes several benchmark datasets after projecting them to their first 2 principal components. In the upper dimensions, i.e., the number of features is greater than 3, the new generated examples seem to be located within the same space of the minority examples with respect to the maximum variance in all of the presented datasets (Figure 8). The



(a) Original data with three outlier examples located in the majority class's boundary (b) Result show that SMOTEFUNA is less sensitive to outlier examples

FIGURE 7. Illustration of how validating the new generated examples with the nearest neighbours makes the algorithm less sensitive to outlier, blue points are the majority.



(a) PCA on D4 (b) PCA on D6 (c) PCA on D12 (d) PCA on D13 (e) PCA on D27 (f) PCA on D31

FIGURE 8. Projection of the minority, majority and the new created examples using 2 PCA components on different datasets. Datasets names correspond to the datasets numbers mentioned in Table 1.

results of the principal component analysis (PCA) projection demonstrate that SMOTEFUNA forces the data to follow the distribution of the minority class and prevents the generation of synthetic examples in an overlapping manner with the majority class.

IV. EXPERIMENTS AND RESULTS

In this section, the proposed algorithm is compared with a subset of the state-of-the-art methods for synthetic oversampling. In particular, we evaluate SMOTEFUNA with respect to SMOTE [14], ADASYN [18], SWIM [19].

In our experiments, we utilize *naive Bayes* (with kernel smoothing density estimate) and *support vector machines* (with error-correcting output codes model) classifiers. The Matlab 2019b environment is used to perform the experiments involving SMOTE and ADASYN, while Python is used for SWIM[1] because there is no Matlab implementation available.

TABLE 1. Dataset description. nInes is the number of inestances, nCl is the number of classes, IR is the Imbalance ratio and nAt is the number of attributes(features).

#	Name	nInes	nCl	IR	nAt
1	Ar1	121	2	13.4	30
2	Ar3	63	2	7.875	30
3	Ar4	107	2	5.45	30
4	Ar5	36	2	4.5	30
5	Ar6	101	2	6.7	30
6	Kc1	2109	2	605	22
7	Kc2	522	2	4.9	22
8	Pc1	1109	2	14.4	22
9	Pc3	1563	2	9.8	38
10	Pc4	1458	2	8.2	38
11	Australian	690	2	1.25	42
12	Bank	1372	2	1.25	4
13	Heart	270	2	1.25	25
14	Oil-Spill	937	2	21.85	49
15	Phoneme	5404	2	2.41	5
16	Abalone19	4174	2	129.44	8
17	Abalone9-18	731	2	16.4	8
18	Page-blocks0	5472	2	8.79	11
19	Pima	768	2	1.87	9
20	Segment0	2308	2	6.2	20
21	Shuttle-c0	1829	2	13.87	10
22	Vehicle0	846	2	3.25	19
23	Vehicle1	846	2	2.9	19
24	Vehicle2	846	2	2.88	19
25	Vehicle3	846	2	2.99	19
26	Vowel0	988	2	9.98	14
27	Wisconsin	683	2	1.86	10
28	Yeast-1-2-8-9	947	2	30.57	9
29	Yeast-1-4-5-8	693	2	22.1	9
30	Yeast1	1484	2	2.46	9
31	Yeast3	1484	2	8.1	9
32	Yeast4	1484	2	28.1	9
33	Yeast5	1484	2	32.73	9

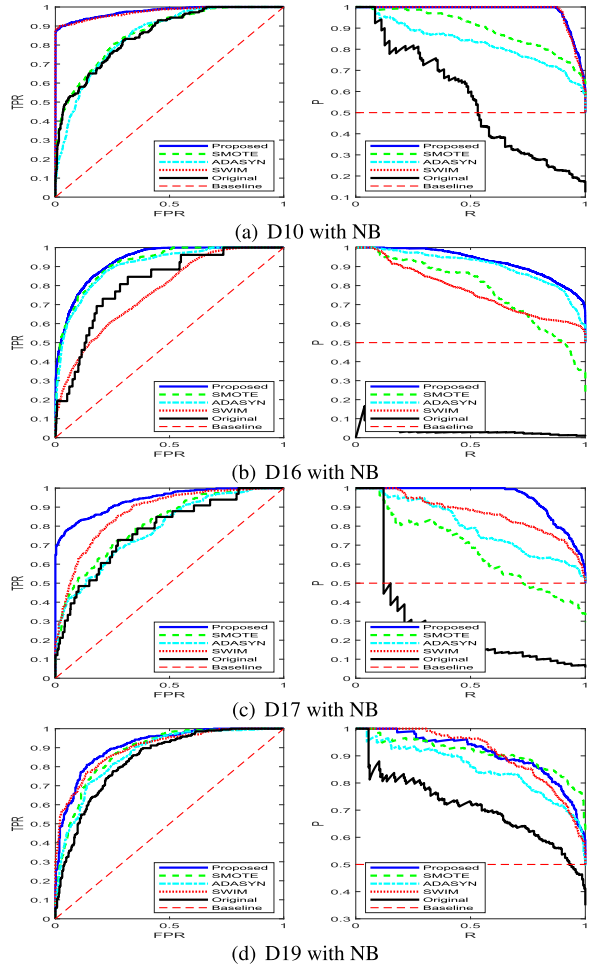
A. EVALUATION MEASURES

To validate the performance of the proposed method, each experiment is conducted using *5-fold* cross-validation. We report the performance in terms of the receiver operating characteristic curve (ROC) and area under the ROC curve (AUC) [39]–[41]. As an additional metric of evaluation, we utilize Precision-Recall curves (PR curve) and the area under PR curve (AUPR). These two measures are commonly used to test the performance of oversampling algorithms and provide an alternate perspective on performance [42]–[44]. ROC and PR curves indicate a good performance when the area under each curve is close to 1. The experiments are

[1] https://github.com/cbellinger27/SWIM_RBF/.

TABLE 2. Number of wins for each algorithm using both classifiers and different evaluation measures, in case of equal performance of two (or more) methods, they both considered as winners.

Classifier	AUC				
	SMOTEFUNA	SMOTE	ADASYN	SWIM	Original Data
SVM	21	3	9	1	7
NB	24	5	2	6	2
Classifier	**AUPR**				
	SMOTEFUNA	SMOTE	ADASYN	SWIM	Original Data
SVM	19	11	7	1	3
NB	20	9	2	6	1



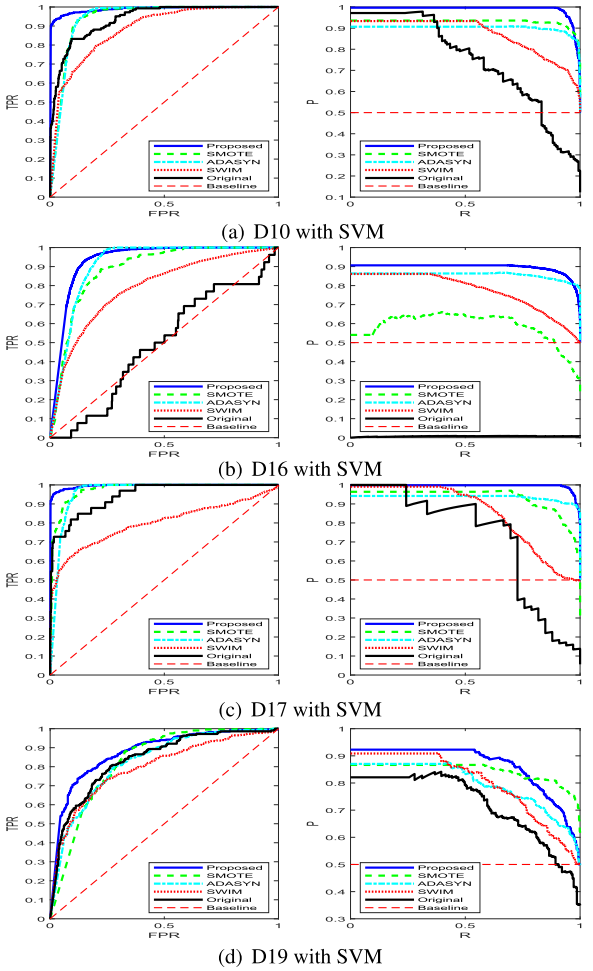
(a) D10 with NB

(b) D16 with NB

(c) D17 with NB

(d) D19 with NB

FIGURE 9. The ROC and PR curves of the proposed method compared to the other methods and the original data using NB classifier.



(a) D10 with SVM

(b) D16 with SVM

(c) D17 with SVM

(d) D19 with SVM

FIGURE 10. The ROC and PR curves of the proposed method compared to the other methods and the original data using SVM classifier.

repeated 30 times using the *5-fold* cross validation approach, and the mean of the performance measures (AUC and AUPR) is considered as the final result for each method on each dataset. Also, the standard deviation of the performance is calculated for each method on each dataset to provide an overall measure of performance.

B. DATASETS

Table 1 outlines the 33 benchmark datasets employed to evaluate the proposed method. These were selected as they vary in their imbalance ratio (IR), the number of instances (nInes) and the number of attributes (nAt), while the number of classes (nCl) is the same (binary classification). IR is calculated as the number of examples in the majority class divided by the number of example in the minority class [45], [46]. In our experiments, each dataset is used considering all of its attributes. Also, each dataset is standardized using *z-score* prior to the learning process

All of the datasets are publicly available to the research community. The first set of datasets can be downloaded

TABLE 3. p-value of the Wilcoxon signed-rank test of SMOTEFUNA vs SMOTE, SMOTEFUNA vs AD, SMOTEFUNA vs SWIM and SMOTEFUNA vs OR, using the AUC and AUPR obtained by SVM and NB classifiers.

Measure	Classifier	SMOTE	AD	SWIM	OR
AUC	SVM	3.5338×10^{-3}	4.4798×10^{-3}	1.0×10^{-6}	1.1395×10^{-3}
	NB	3.05×10^{-5}	1.9×10^{-6}	7.632×10^{-4}	5.5×10^{-6}
AUPR	SVM	3.6234×10^{-2}	1.471×10^{-4}	1.0×10^{-6}	2.9×10^{-6}
	NB	2.7350×10^{-2}	1.5×10^{-6}	6.332×10^{-3}	8×10^{-7}

TABLE 4. Wilcoxon *R*, W+, and W−, separated by "/".

Measure	Classifier	Smote	AD	SWIM	OR
AUC	SVM	0.51/ 420/108	0.49/ 416/112	0.85/ 526/2	0.57/ 438/90
	NB	0.73/ 487/41	0.83/ 519/9	0.59/ 444/84	0.79/ 507/21
AUPR	SVM	0.36/ 376/152	0.66/ 467/61	0.85/ 526/2	0.81/ 514/14
	NB	0.38/ 382/146	0.84/ 521/7	0.48/ 410/118	0.86/ 528/0

TABLE 5. AUC results of different methods using SVM classifier, mean ± Standard deviation (STD) calculated over 30 trials.

Dataset	SMOTEFUNA	SMOTE	ADASYN	SWIM	OR
	Mean ± STD	Mean ± STD	Mean ± STD	Mean ± STD	Mean ± STD
D1	**0.9858 ± 0.0017**	0.9501 ± 0.0081	0.9750 ± 0.0022	0.9236 ± 0.0027	0.9117 ± 0.0111
D2	0.9897 ± 0.0013	0.9710 ± 0.0072	**0.9924 ± 0.0030**	0.9851 ± 0.0035	0.9343 ± 0.0075
D3	**0.9729 ± 0.0021**	0.9283 ± 0.0043	0.9266 ± 0.0054	0.9285 ± 0.0042	0.8922 ± 0.0068
D4	0.9998 ± 0.0006	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	0.9988 ± 0.0003	**1.0000 ± 0.0000**
D5	**0.9830 ± 0.0012**	0.9350 ± 0.0060	0.9377 ± 0.0062	0.9143 ± 0.0042	0.9059 ± 0.0066
D6	0.9994 ± 0.0000	0.9998 ± 0.0001	**1.0000 ± 0.0000**	0.9974 ± 0.0001	0.9999 ± 0.0000
D7	**0.9926 ± 0.0003**	0.9871 ± 0.0012	0.9855 ± 0.0007	0.9502 ± 0.0006	0.9741 ± 0.0008
D8	**0.9994 ± 0.0001**	0.9991 ± 0.0001	0.9990 ± 0.0001	0.9984 ± 0.0001	**0.9994 ± 0.0001**
D9	**0.9623 ± 0.0012**	0.8647 ± 0.0012	0.8577 ± 0.0014	0.8012 ± 0.0004	0.7708 ± 0.0106
D10	**0.9890 ± 0.0002**	0.9365 ± 0.0008	0.9355 ± 0.0012	0.8777 ± 0.0005	0.9267 ± 0.0010
D11	**0.9457 ± 0.0012**	0.9291 ± 0.0021	0.9169 ± 0.0030	0.9266 ± 0.0015	0.9373 ± 0.0015
D12	0.9929 ± 0.0012	0.9985 ± 0.0006	**0.9997 ± 0.0000**	0.9826 ± 0.0002	**0.9997 ± 0.0000**
D13	**0.9492 ± 0.0016**	0.9442 ± 0.0022	0.9323 ± 0.0025	0.9313 ± 0.0016	0.9373 ± 0.0016
D14	**0.9936 ± 0.0007**	0.9880 ± 0.0006	0.9847 ± 0.0010	0.9266 ± 0.0004	0.9670 ± 0.0025
D15	**0.8488 ± 0.0007**	0.7328 ± 0.0008	0.7746 ± 0.0005	0.7910 ± 0.0001	0.8023 ± 0.0001
D16	**0.9327 ± 0.0020**	0.8962 ± 0.0017	0.9036 ± 0.0013	0.7870 ± 0.0001	0.5821 ± 0.0626
D17	**0.9840 ± 0.0016**	0.9719 ± 0.0008	0.9670 ± 0.0008	0.8059 ± 0.0003	0.9546 ± 0.0015
D18	**0.9906 ± 0.0001**	0.9598 ± 0.0005	0.9472 ± 0.0007	0.8024 ± 0.0001	0.9297 ± 0.0010
D19	**0.8822 ± 0.0015**	0.8335 ± 0.0014	0.8211 ± 0.0015	0.8050 ± 0.0006	0.8396 ± 0.0005
D20	0.9995 ± 0.0000	**0.9998 ± 0.0000**	0.9994 ± 0.0001	0.9951 ± 0.0000	0.9996 ± 0.0000
D21	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
D22	0.9974 ± 0.0001	0.9943 ± 0.0005	0.9937 ± 0.0006	0.9568 ± 0.0004	**0.9975 ± 0.0002**
D23	0.8613 ± 0.0031	0.8807 ± 0.0014	0.8801 ± 0.0019	0.7753 ± 0.0005	**0.8839 ± 0.0007**
D24	0.9732 ± 0.0011	0.9919 ± 0.0004	**0.9923 ± 0.0004**	0.9055 ± 0.0005	0.9916 ± 0.0003
D25	0.8159 ± 0.0038	0.8723 ± 0.0017	**0.8769 ± 0.0017**	0.7569 ± 0.0008	0.8667 ± 0.0006
D26	0.9814 ± 0.0011	0.9936 ± 0.0004	0.9942 ± 0.0006	0.9569 ± 0.0003	**0.9946 ± 0.0003**
D27	**0.9927 ± 0.0004**	0.9898 ± 0.0004	0.9867 ± 0.0005	0.9899 ± 0.0004	0.9921 ± 0.0004
D28	0.9997 ± 0.0001	0.9999 ± 0.0000	**1.0000 ± 0.0000**	0.9999 ± 0.0000	0.9976 ± 0.0003
D29	0.9991 ± 0.0003	0.9995 ± 0.0002	**0.9998 ± 0.0002**	0.9988 ± 0.0001	0.9953 ± 0.0004
D30	**0.8987 ± 0.0007**	0.7741 ± 0.0008	0.7703 ± 0.0010	0.7600 ± 0.0002	0.7943 ± 0.0004
D31	**0.9856 ± 0.0004**	0.9669 ± 0.0005	0.9549 ± 0.0007	0.9530 ± 0.0002	0.9693 ± 0.0002
D32	**0.9828 ± 0.0014**	0.9183 ± 0.0014	0.9160 ± 0.0017	0.8712 ± 0.0002	0.8218 ± 0.0279
D33	**0.9944 ± 0.0003**	0.9870 ± 0.0003	0.9866 ± 0.0003	0.9791 ± 0.0002	0.9883 ± 0.0002

from the PROMISE repository [47], and the second set of datasets is available from KEEL website (Imbalance ratio between 1.5 and 9 and imbalance ratio higher than 9 - Part I) [48], [49].

TABLE 6. AUPR results of different methods using SVM classifier, mean ± standard deviation (STD) calculated over 30 trials.

Dataset	SMOTEFUNA	SMOTE	ADASYN	SWIM	OR
	Mean ± STD	Mean ± STD	Mean ± STD	Mean ± STD	Mean ± STD
D1	**0.9800 ± 0.0028**	0.8106 ± 0.0192	0.9500 ± 0.0044	0.9066 ± 0.0054	0.7047 ± 0.0262
D2	**0.9918 ± 0.0016**	0.9442 ± 0.0128	0.9864 ± 0.0062	0.9787 ± 0.0071	0.8508 ± 0.0176
D3	**0.9795 ± 0.0021**	0.8753 ± 0.0111	0.8844 ± 0.0098	0.9305 ± 0.0053	0.7917 ± 0.0105
D4	0.9998 ± 0.0005	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	0.9988 ± 0.0003	**1.0000 ± 0.0000**
D5	**0.9874 ± 0.0008**	0.8755 ± 0.0123	0.9072 ± 0.0119	0.9267 ± 0.0042	0.7931 ± 0.0125
D6	0.9988 ± 0.0001	**0.9999 ± 0.0001**	**0.9999 ± 0.0001**	0.9962 ± 0.0002	0.9995 ± 0.0000
D7	**0.9939 ± 0.0005**	0.9787 ± 0.0024	0.9724 ± 0.0015	0.9420 ± 0.0016	0.9604 ± 0.0016
D8	**0.9989 ± 0.0002**	0.9975 ± 0.0004	0.9981 ± 0.0003	0.9974 ± 0.0002	0.9849 ± 0.0034
D9	**0.9670 ± 0.0013**	0.8454 ± 0.0016	0.8050 ± 0.0020	0.7785 ± 0.0008	0.3911 ± 0.0096
D10	**0.9893 ± 0.0003**	0.9221 ± 0.0011	0.8933 ± 0.0022	0.8593 ± 0.0015	0.7212 ± 0.0018
D11	0.9279 ± 0.0025	**0.9306 ± 0.0025**	0.8857 ± 0.0054	0.8996 ± 0.0028	0.9027 ± 0.0037
D12	0.9906 ± 0.0026	0.9988 ± 0.0005	**0.9996 ± 0.0000**	0.9885 ± 0.0003	**0.9996 ± 0.0000**
D13	0.9374 ± 0.0029	**0.9451 ± 0.0030**	0.9128 ± 0.0038	0.9236 ± 0.0033	0.9059 ± 0.0025
D14	**0.9894 ± 0.0014**	0.9462 ± 0.0031	0.9702 ± 0.0019	0.9218 ± 0.0008	0.7763 ± 0.0069
D15	0.7980 ± 0.0012	**0.9372 ± 0.0002**	0.7159 ± 0.0009	0.7334 ± 0.0002	0.5662 ± 0.0006
D16	**0.8987 ± 0.0028**	0.5777 ± 0.0049	0.8414 ± 0.0022	0.7720 ± 0.0003	0.0140 ± 0.0036
D17	**0.9845 ± 0.0015**	0.9213 ± 0.0025	0.9475 ± 0.0015	0.8529 ± 0.0006	0.7104 ± 0.0038
D18	**0.9908 ± 0.0001**	0.9827 ± 0.0002	0.9263 ± 0.0009	0.8651 ± 0.0001	0.7921 ± 0.0012
D19	**0.8645 ± 0.0027**	0.8519 ± 0.0018	0.7841 ± 0.0026	0.8004 ± 0.0016	0.7302 ± 0.0017
D20	0.9993 ± 0.0000	**0.9998 ± 0.0000**	0.9988 ± 0.0001	0.9951 ± 0.0001	0.9958 ± 0.0001
D21	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
D22	**0.9957 ± 0.0003**	0.9930 ± 0.0006	0.9885 ± 0.0011	0.9483 ± 0.0010	0.9914 ± 0.0013
D23	0.8599 ± 0.0040	**0.8840 ± 0.0015**	0.8164 ± 0.0038	0.8019 ± 0.0020	0.7237 ± 0.0022
D24	0.9628 ± 0.0016	**0.9916 ± 0.0004**	0.9868 ± 0.0009	0.9043 ± 0.0010	0.9697 ± 0.0015
D25	0.8140 ± 0.0045	**0.8739 ± 0.0019**	0.8144 ± 0.0032	0.7460 ± 0.0021	0.6464 ± 0.0025
D26	0.9672 ± 0.0023	0.9849 ± 0.0010	**0.9889 ± 0.0012**	0.9361 ± 0.0007	0.9432 ± 0.0044
D27	0.9859 ± 0.0008	**0.9877 ± 0.0006**	0.9740 ± 0.0011	0.9807 ± 0.0008	0.9728 ± 0.0013
D28	0.9994 ± 0.0003	0.9998 ± 0.0003	**1.0000 ± 0.0001**	0.9999 ± 0.0000	0.9315 ± 0.0059
D29	0.9982 ± 0.0006	0.9973 ± 0.0015	**0.9995 ± 0.0004**	0.9980 ± 0.0002	0.9134 ± 0.0065
D30	**0.8916 ± 0.0010**	0.8699 ± 0.0007	0.7365 ± 0.0012	0.7510 ± 0.0005	0.5878 ± 0.0009
D31	**0.9756 ± 0.0008**	0.9578 ± 0.0008	0.9248 ± 0.0014	0.9377 ± 0.0004	0.7953 ± 0.0026
D32	**0.9758 ± 0.0023**	0.8236 ± 0.0026	0.8884 ± 0.0018	0.8576 ± 0.0005	0.2615 ± 0.0333
D33	**0.9893 ± 0.0006**	0.9387 ± 0.0016	0.9739 ± 0.0005	0.9617 ± 0.0003	0.6909 ± 0.0098

C. RESULTS

ROC curve provides an efficient evaluation of the learning process as it represents the relation between the two important measures, true positive rate (TPR) and false positive rate (FPR). ROC curve is a plot of TPR as a function of FPR at different performance thresholds. Figures 9 and 10 compare the ROC and PR curves of different methods to SMOTEFUNA using several datasets and classifiers using the first fold; the averages using all folds (all trials) are provided in the tables at Appendix.

As one can see in Figures 9 and 10, the performance of the proposed method is better than those of the other methods. The performance curves of SMOTE and ADASYN follow very similar trajectories, which indicates that they cause the induced classifier to behave similarly. Notably, SMOTE and ADASYN do not provide any significant improvement with respect to the ROC curves on some datasets. Oppositely, SMOTEFUNA does improve performance in these cases. In the case of SWIM, the performance is alternating. Although it works well on some datasets, it seems to not obtain satisfactory results on the majority of the datasets. This is a noteworthy result as it was initially proposed for extreme levels of imbalance. Our results suggest that under certain conditions, it may perform very well on less extreme levels of imbalance.

AUC provides an efficient evaluation of the performance. The perfect performance of a classifier on a dataset is when AUC is equal to 1. Due to difficulty of presenting all ROC and PR curves for all the datasets, we use AUC and AUPR to give a clear vision of the performance of different methods using these two measures. The complete tables can be found in Appendix. Table 2 summarizes the number of wins for SMOTEFUNA algorithm on all of the used datasets, compared to the other algorithms.

TABLE 7. AUC results of different methods using NB classifier, Mean ± Standard deviation (STD) calculated over 30 trials.

Dataset	SMOTEFUNA	cSMOTE	ADASYN	SWIM	OR
	Mean ± STD	Mean ± STD	Mean ± STD	Mean ± STD	Mean ± STD
D1	0.9817 ± 0.0017	0.9345 ± 0.0151	0.9301 ± 0.0110	**0.9830 ± 0.0004**	0.9237 ± 0.0076
D2	**0.9903 ± 0.0009**	0.9607 ± 0.0067	0.9660 ± 0.0036	0.9876 ± 0.0004	0.9414 ± 0.0046
D3	**0.9750 ± 0.0009**	0.9006 ± 0.0036	0.9042 ± 0.0045	0.9728 ± 0.0004	0.8993 ± 0.0024
D4	**0.9972 ± 0.0004**	0.9916 ± 0.0017	0.9936 ± 0.0013	0.9944 ± 0.0005	0.9876 ± 0.0012
D5	**0.9790 ± 0.0019**	0.9035 ± 0.0096	0.9352 ± 0.0077	0.9637 ± 0.0003	0.8641 ± 0.0083
D6	**0.9977 ± 0.0001**	0.9921 ± 0.0001	0.9902 ± 0.0001	0.9935 ± 0.0000	0.9828 ± 0.0001
D7	**0.9776 ± 0.0011**	0.9256 ± 0.0023	0.9031 ± 0.0031	0.9751 ± 0.0001	0.9601 ± 0.0008
D8	**0.9998 ± 0.0000**	0.9974 ± 0.0001	0.9979 ± 0.0001	0.9988 ± 0.0000	0.9921 ± 0.0001
D9	0.9697 ± 0.0005	0.8358 ± 0.0013	0.8268 ± 0.0014	**0.9720 ± 0.0002**	0.8091 ± 0.0005
D10	**0.9747 ± 0.0004**	0.8624 ± 0.0024	0.8488 ± 0.0034	0.9718 ± 0.0003	0.8423 ± 0.0022
D11	**0.9416 ± 0.0008**	0.9400 ± 0.0017	0.9186 ± 0.0028	0.9351 ± 0.0007	0.9321 ± 0.0008
D12	0.9786 ± 0.0003	**0.9796 ± 0.0001**	0.9787 ± 0.0004	0.9726 ± 0.0001	0.9786 ± 0.0001
D13	0.9252 ± 0.0069	**0.9383 ± 0.0017**	0.9137 ± 0.0038	0.8698 ± 0.0107	0.9301 ± 0.0011
D14	**0.9946 ± 0.0001**	0.9748 ± 0.0022	0.9774 ± 0.0016	0.9924 ± 0.0000	0.9536 ± 0.0014
D15	**0.8871 ± 0.0012**	0.8636 ± 0.0001	0.8268 ± 0.0004	0.8650 ± 0.0000	0.8584 ± 0.0001
D16	0.9098 ± 0.0036	**0.9102 ± 0.0025**	0.9041 ± 0.0023	0.7682 ± 0.0000	0.8344 ± 0.0030
D17	**0.9454 ± 0.0028**	0.8005 ± 0.0036	0.7940 ± 0.0048	0.8678 ± 0.0002	0.7766 ± 0.0019
D18	0.9799 ± 0.0001	0.9816 ± 0.0001	0.9590 ± 0.0004	**0.9837 ± 0.0000**	**0.9837 ± 0.0002**
D19	**0.9119 ± 0.0014**	0.8750 ± 0.0035	0.8548 ± 0.0049	0.8969 ± 0.0008	0.8410 ± 0.0041
D20	**0.9996 ± 0.0000**	0.9995 ± 0.0000	0.9986 ± 0.0001	0.9937 ± 0.0000	0.9994 ± 0.0000
D21	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
D22	**0.9702 ± 0.0008**	0.9605 ± 0.0009	0.9231 ± 0.0015	0.9326 ± 0.0006	0.9401 ± 0.0007
D23	0.8405 ± 0.0031	0.7741 ± 0.0009	0.7499 ± 0.0015	**0.8501 ± 0.0005**	0.7575 ± 0.0007
D24	**0.9859 ± 0.0008**	0.9676 ± 0.0010	0.9635 ± 0.0023	0.9552 ± 0.0003	0.9598 ± 0.0008
D25	0.8077 ± 0.0021	0.7683 ± 0.0011	0.7492 ± 0.0011	**0.8498 ± 0.0003**	0.7527 ± 0.0007
D26	0.9930 ± 0.0004	**0.9980 ± 0.0003**	0.9962 ± 0.0007	0.9959 ± 0.0000	0.9964 ± 0.0002
D27	**0.9973 ± 0.0001**	0.9955 ± 0.0003	0.9899 ± 0.0005	0.9963 ± 0.0000	0.9961 ± 0.0001
D28	**1.0000 ± 0.0000**	0.9999 ± 0.0000	**1.0000 ± 0.0000**	0.9991 ± 0.0000	0.9991 ± 0.0001
D29	**0.9999 ± 0.0000**	0.9993 ± 0.0002	0.9996 ± 0.0002	0.9987 ± 0.0000	0.9986 ± 0.0001
D30	**0.8511 ± 0.0036**	0.7383 ± 0.0012	0.7111 ± 0.0020	0.4923 ± 0.0009	0.7187 ± 0.0012
D31	**0.9947 ± 0.0003**	0.9699 ± 0.0055	0.9641 ± 0.0022	0.9386 ± 0.0041	0.9806 ± 0.0021
D32	**0.9869 ± 0.0019**	0.9427 ± 0.0015	0.9407 ± 0.0017	0.8959 ± 0.0054	0.9207 ± 0.0045
D33	**0.9986 ± 0.0002**	0.9954 ± 0.0006	0.9957 ± 0.0003	0.9880 ± 0.0006	0.9936 ± 0.0002

As presented in Table 2, SMOTEFUNA has the highest number of wins, compared to other resampling methods, followed by SMOTE and ADASYN, respectively. Finally, the baseline classifier, without resampling (original data), has a higher number of wins than SWIM, in term of AUC and AUPR values, when SVM is used; and slightly better than those of the original data when NB is used. The aggregated results in this table suggest that SMOTEFUNA is a reliable alternative solution for oversampling imbalance data.

Tables 5 and 7 illustrate the performance of SMOTEFUNA in terms of AUC compared to other methods using SVM and NB, respectively. In general, the results are better than that of the other methods on most of the tested datasets. In some cases, the results were not satisfactory. This is perhaps due to a specific type of outliers, where an outlier example(s) is far from the minority data and even farther from the majority data. In such a case, SMOTEFUNA might add examples in the space between the minority and the outlier example. Such situations need to be investigated in future studies in order to improve the performance on machine learning datasets, where this phenomenon is more likely to occur. Using SVM classifier, SMOTEFUNA records higher results on 21 datasets out of 33, while it records better results on 24 datasets using NB, in terms of AUC.

Tables 6 and 8 present the results of AUPR on the tested datasets using the same classifiers and the same validation approach. For AUPR, SMOTEFUNA is better than the other methods compared on 19 datasets using SVM, and on 20 datasets when NB is used.

As can be clearly noticed from the results of AUC and AUPR, the learning process is improved after oversampling using SMOTEFUNA. For example, the AUPR result using the original data of D10 (see Table 8) has improved from around **53%** to **98%**, and from around **29%** to **95%** when

TABLE 8. AUPR results of different methods using NB classifier, Mean ± Standard deviation (STD) calculated over 30 trials.

Dataset	SMOTEFUNA	SMOTE	ADASYN	SWIM	OR
	Mean ± STD	Mean ± STD	Mean ± STD	Mean ± STD	Mean ± STD
D1	0.9785 ± 0.0027	0.7444 ± 0.0368	0.9274 ± 0.0118	**0.9870 ± 0.0003**	0.5987 ± 0.0335
D2	**0.9924 ± 0.0006**	0.9202 ± 0.0123	0.9705 ± 0.0032	0.9907 ± 0.0002	0.8410 ± 0.0059
D3	0.9783 ± 0.0016	0.8304 ± 0.0068	0.9041 ± 0.0049	**0.9788 ± 0.0003**	0.7549 ± 0.0071
D4	**0.9974 ± 0.0004**	0.9867 ± 0.0027	0.9941 ± 0.0012	0.9951 ± 0.0004	0.9665 ± 0.0028
D5	**0.9839 ± 0.0015**	0.8306 ± 0.0134	0.9443 ± 0.0062	0.9739 ± 0.0002	0.6944 ± 0.0157
D6	**0.9977 ± 0.0001**	0.9962 ± 0.0000	0.9892 ± 0.0001	0.9939 ± 0.0000	0.9190 ± 0.0002
D7	**0.9822 ± 0.0007**	0.9399 ± 0.0017	0.9216 ± 0.0024	0.9808 ± 0.0001	0.8963 ± 0.0019
D8	**0.9998 ± 0.0001**	0.9945 ± 0.0001	0.9972 ± 0.0001	0.9988 ± 0.0000	0.8982 ± 0.0014
D9	0.9692 ± 0.0010	0.8708 ± 0.0012	0.8381 ± 0.0018	**0.9753 ± 0.0004**	0.4509 ± 0.0026
D10	**0.9795 ± 0.0006**	0.8914 ± 0.0023	0.8285 ± 0.0042	0.9784 ± 0.0002	0.5333 ± 0.0048
D11	0.9527 ± 0.0011	**0.9674 ± 0.0012**	0.9325 ± 0.0024	0.9476 ± 0.0011	0.9340 ± 0.0015
D12	0.9774 ± 0.0004	**0.9901 ± 0.0001**	0.9707 ± 0.0010	0.9716 ± 0.0001	0.9703 ± 0.0002
D13	0.9255 ± 0.0058	**0.9537 ± 0.0015**	0.9118 ± 0.0037	0.8908 ± 0.0067	0.9087 ± 0.0016
D14	**0.9957 ± 0.0001**	0.9424 ± 0.0043	0.9753 ± 0.0016	0.9944 ± 0.0000	0.7050 ± 0.0057
D15	0.8554 ± 0.0024	**0.9740 ± 0.0000**	0.7851 ± 0.0007	0.8477 ± 0.0001	0.6744 ± 0.0005
D16	**0.9013 ± 0.0041**	0.7636 ± 0.0087	0.9012 ± 0.0028	0.7724 ± 0.0001	0.0683 ± 0.0084
D17	**0.9529 ± 0.0032**	0.6738 ± 0.0041	0.8029 ± 0.0037	0.8693 ± 0.0003	0.2874 ± 0.0031
D18	0.9847 ± 0.0001	**0.9951 ± 0.0000**	0.9531 ± 0.0004	0.9874 ± 0.0000	0.9027 ± 0.0007
D19	**0.9144 ± 0.0015**	0.9082 ± 0.0030	0.8381 ± 0.0058	0.9064 ± 0.0009	0.7304 ± 0.0074
D20	0.9996 ± 0.0000	**0.9998 ± 0.0000**	0.9985 ± 0.0001	0.9940 ± 0.0000	0.9970 ± 0.0001
D21	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**	**1.0000 ± 0.0000**
D22	**0.9727 ± 0.0007**	0.9696 ± 0.0008	0.9212 ± 0.0020	0.9336 ± 0.0006	0.8156 ± 0.0030
D23	0.8083 ± 0.0041	0.8605 ± 0.0005	0.7534 ± 0.0036	**0.8687 ± 0.0004**	0.5829 ± 0.0012
D24	**0.9879 ± 0.0005**	0.9793 ± 0.0005	0.9619 ± 0.0024	0.9596 ± 0.0003	0.9012 ± 0.0014
D25	0.7747 ± 0.0040	0.8447 ± 0.0005	0.7437 ± 0.0015	**0.8649 ± 0.0003**	0.5479 ± 0.0014
D26	0.9929 ± 0.0004	**0.9975 ± 0.0003**	0.9963 ± 0.0008	0.9961 ± 0.0000	0.9698 ± 0.0013
D27	**0.9973 ± 0.0001**	0.9972 ± 0.0002	0.9887 ± 0.0005	0.9962 ± 0.0001	0.9927 ± 0.0001
D28	**1.0000 ± 0.0000**	0.9995 ± 0.0002	**1.0000 ± 0.0000**	0.9989 ± 0.0000	0.9667 ± 0.0028
D29	**0.9999 ± 0.0000**	0.9972 ± 0.0007	0.9995 ± 0.0002	0.9983 ± 0.0000	0.9601 ± 0.0039
D30	0.8609 ± 0.0036	**0.8659 ± 0.0009**	0.6945 ± 0.0024	0.5719 ± 0.0008	0.4924 ± 0.0018
D31	**0.9946 ± 0.0004**	0.9745 ± 0.0047	0.9576 ± 0.0026	0.9373 ± 0.0042	0.8621 ± 0.0129
D32	**0.9877 ± 0.0022**	0.8781 ± 0.0023	0.9302 ± 0.0020	0.8985 ± 0.0054	0.3308 ± 0.0093
D33	**0.9983 ± 0.0003**	0.9851 ± 0.0026	0.9945 ± 0.0004	0.9846 ± 0.0009	0.8190 ± 0.0061

learning from D17, which can be considered as a significant enhancement of the learning process. The rest of the comparative results are presented in Tables 5–8.

D. STATISTICAL ANALYSIS

To further analyze the results shown in Tables 5, 6, 7 and 8, and to see if there are any significant differences between the proposed SMOTEFUNA and the other methods compared, a Wilcoxon signed-rank test is performed on the SMOTEFUNA results versus each other method, using SVM and NB classifiers on both measures (AUC and AUPR) over all of the 33 datasets tested. We used *SPSS* (version 24) software for this purpose, the results are recorded in Tables 3 and 4, these results are further verified using an online statistical tool.[2]

[2]http://www.statskingdom.com/175wilcoxon_signed_ranks.html

The Wilcoxon test is suitable for our comparison because it is safer than parametric tests (such as t-test) since it does not assume homogeneity or normal distribution of data. Therefore, it can be applied to error ratios, classification results (accuracy, precision, recall), or any other classifier evaluation measure [50]. The Wilcoxon test aims to find if a null hypothesis is true or not. The null hypothesis H_0 assumes that there is no significant difference between the classification results (observations) obtained from two different methods. We assume that the null hypothesis is rejected if the p-value of the Wilcoxon test is less than $\alpha = 0.05$.

As can be seen from Table 3, the p-values for all the paired tests are less than 0.05, and therefore, the H_0 is rejected for all the paired tests. This shows that the differences between the (AUC and AUPR) results of SMOTEFUNA and that of the other methods compared are big enough to be statistically significant.

However, the p-value does not provide information about the relationship strength between variables, and does not allow us to determine which variable dominates the others, i.e., is better [51]. Thus, based solely on the p-values shown in Table 3, we cannot tell whether these significant results are in favor of SMOTEFUNA or not, as they have only shown that there is a significant difference between the results of SMOTEFUNA and each of the other methods tested.

Therefore, in order to see which method performs better, we need to look at other measures associated with the Wilcoxon test, namely, sum of the positive ranks ($W+$), sum of the negative ranks ($W-$), and the effect size (R), which serve such purpose well. The effect sizes of the Wilcoxon signed-rank test can be calculated using

$$R = \frac{|Z|}{\sqrt{N}},$$

where Z is the test statistic, and N the total number of paired samples, which is equal to 33 in our case. The sizes of effects can be categorized as small ($R \leq 0.1$), medium ($0.1 < R < 0.5$), or large ($R \geq 0.5$) [52]. The three mentioned measures are shown in Table 4, which were obtained by the same Wilcoxon test, whose p-values are shown in Table 3.

The results in Table 4 show the magnitude of effects obtained by the comparison of SMOTEFUNA with the other methods. Since the results of the other methods were subtracted from those of the SMOTEFUNA, $W+$ is counted for SMOTEFUNA, while W- is counted for the other method compared. The effect size is considered as large in most cases (12 cases) and medium in the remaining 4 cases. Another interesting note is that $W+$ is significantly higher than $W-$ for all the paired tests, i.e. SMOTEFUNA is the winner of most datasets, regardless the classifier or the measure used. This proves that the significant difference shown by the p-values is in the favor of our proposed SMOTEFUNA method in all cases (two different classifiers and two different measures).

V. CONCLUSION

In this paper, we present a novel, parameter-free algorithm for performing synthetic oversampling on imbalanced data sets, for the purpose of better learning from imbalanced data. The proposed method, SMOTEFUNA, is evaluated on 33 benchmark machine learning datasets using two different classifiers. The experimental results show that, in general, SMOTEFUNA outperforms SMOTE, ADASYN and SWIM in terms of ROC and PR curves as well as AUC and AUPR values.

Our analysis reveals that, although SMOTEFUNA typically performs well, it may be negatively impacted, particularly when an outlier example of minority class is isolated from both the minority and majority classes. In such a case, the potential spaces of the new examples will be shifted away from the ideal decision boundary, and towards the outlier example. In future implementations, this can be managed by detecting and removing the outliers prior to utilizing SMOTEFUNA [53]–[56]. Alternatively, the outlier problem can be alleviated using a distance metric that is not affected by outliers, such as [57], [58].

Our future efforts will include further evaluation of SMOTEFUNA compared to a larger number of related methods, and extending of SMOTEFUNA to work on multi-class imbalanced data, nominal and big datasets.

APPENDIX

A. TABULAR RESULTS

See Tables 5–8

ACKNOWLEDGMENT

A. S. Tarawneh would like to thank the Tempus Public Foundation TPF for sponsoring his Ph.D. study and SZTAKI for providing the required facilities.

REFERENCES

[1] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst. Appl.*, vol. 73, pp. 220–239, May 2017.
[2] N. V. Chawla, "Data mining for imbalanced datasets: An overview," in *Data Mining and Knowledge Discovery Handbook*. Boston, MA, USA: Springer, 2009, pp. 875–886.
[3] M. Fatima and M. Pasha, "Survey of machine learning algorithms for disease diagnostic," *J. Intell. Learn. Syst. Appl.*, vol. 9, no. 1, pp. 1–16, 2017.
[4] U. Fiore, A. De Santis, F. Perla, P. Zanetti, and F. Palmieri, "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection," *Inf. Sci.*, vol. 479, pp. 448–455, Apr. 2019.
[5] R. Pearson, G. Goney, and J. Shwaber, "Imbalanced clustering for microarray time-series," in *Proc. ICML*, vol. 3, 2003, pp. 1–8.
[6] A. S. Tarawneh, A. B. Hassanat, D. Chetverikov, I. Lendak, and C. Verma, "Invoice classification using deep features and machine learning techniques," in *Proc. IEEE Jordan Int. Joint Conf. Elect. Eng. Inf. Technol. (JEEIT)*, Apr. 2019, pp. 855–859.
[7] A. S. Tarawneh, A. B. Hassanat, C. Celik, D. Chetverikov, M. S. Rahman, and C. Verma, "Deep face image retrieval: A comparative study with dictionary learning," in *Proc. 10th Int. Conf. Inf. Commun. Syst. (ICICS)*, Jun. 2019, pp. 185–192.
[8] A. S. Tarawneh, C. Celik, A. B. Hassanat, and D. Chetverikov, "Detailed investigation of deep features with sparse representation and dimensionality reduction in CBIR: A comparative study," *Intell. Data Anal.*, vol. 24, no. 1, pp. 47–68, Feb. 2020.
[9] M. Z. Al-Shamaileh, A. B. Hassanat, A. S. Tarawneh, M. Sohel Rahman, C. Celik, and M. Jawthari, "New Online/Offline text-dependent arabic handwriting dataset for writer authentication and identification," in *Proc. 10th Int. Conf. Inf. Commun. Syst. (ICICS)*, Jun. 2019, pp. 116–121.
[10] C. Bellinger, C. Drummond, and N. Japkowicz, "Manifold-based synthetic oversampling with manifold conformance estimation," *Mach. Learn.*, vol. 107, no. 3, pp. 605–637, Mar. 2018.
[11] S. Wang, W. Jiang, and K.-L. Tsui, "Adjusted support vector machines based on a new loss function," *Ann. Oper. Res.*, vol. 174, no. 1, pp. 83–101, Feb. 2010.
[12] M. A. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," in *Proc. Workshop Learn. Imbalanced Data Sets II (ICML)*, vol. 2, 2003, pp. 1–2.
[13] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 1–6, 2004.
[14] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.
[15] D. Elreedy and A. F. Atiya, "A comprehensive analysis of synthetic minority oversampling technique (SMOTE) for handling class imbalance," *Inf. Sci.*, vol. 505, pp. 32–64, Dec. 2019.

[16] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discovery*. Berlin, Germany: Springer, 2003, pp. 107–119.

[17] T. Pan, J. Zhao, W. Wu, and J. Yang, "Learning imbalanced datasets based on SMOTE and Gaussian distribution," *Inf. Sci.*, vol. 512, pp. 1214–1233, Feb. 2020.

[18] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intelligence)*, Jun. 2008, pp. 1322–1328.

[19] C. Bellinger, S. Sharma, N. Japkowicz, and O. R. Zaïane, "Framework for extreme imbalance classification: SWIM—Sampling with the majority class," *Knowl. Inf. Syst.*, vol. 62, pp. 841–866, Jul. 2019.

[20] W. Siriseriwan and K. Sinapiromsaran, "Adaptive neighbor synthetic minority oversampling technique under 1NN outcast handling," *Songklanakarin J. Sci. Technol*, vol. 39, pp. 565–576, Sep. 2017.

[21] G. Kovács, "An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets," *Appl. Soft Comput.*, vol. 83, Oct. 2019, Art. no. 105662.

[22] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. data Eng.*, vol. 21, no. 9, pp. 1263–1284, Jun. 2009.

[23] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera, "EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling," *Pattern Recognit.*, vol. 46, no. 12, pp. 3460–3471, Dec. 2013.

[24] C. Beyan and R. Fisher, "Classifying imbalanced data sets using similarity based hierarchical decomposition," *Pattern Recognit.*, vol. 48, no. 5, pp. 1653–1672, May 2015.

[25] F. Ren, P. Cao, W. Li, D. Zhao, and O. Zaiane, "Ensemble based adaptive over-sampling method for imbalanced data learning in computer aided detection of microaneurysm," *Computerized Med. Imag. Graph.*, vol. 55, pp. 54–67, Jan. 2017.

[26] S. Chen, H. He, and E. A. Garcia, "RAMOBoost: Ranked minority oversampling in boosting," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1624–1642, Oct. 2010.

[27] J. Xie, M. Hao, W. Liu, and Y. Lin, "Fused variable screening for massive imbalanced data," *Comput. Statist. Data Anal.*, vol. 141, pp. 94–108, Jan. 2020.

[28] T. Zhu, Y. Lin, and Y. Liu, "Improving interpolation-based oversampling for imbalanced data learning," *Knowl.-Based Syst.*, vol. 187, Jan. 2020, Art. no. 104826.

[29] H. Ghaderi Zefrehi and H. Altınçay, "Imbalance learning using heterogeneous ensembles," *Expert Syst. Appl.*, vol. 142, Mar. 2020, Art. no. 113005.

[30] S. Tyagi and S. Mittal, "Sampling approaches for imbalanced data classification problem in machine learning," in *Proc. ICRIC*. Cham, Switzerland: Springer, 2020, pp. 209–221.

[31] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Proc. Int. Conf. Intell. Comput.* Berlin, Germany: Springer, 2005, pp. 878–887.

[32] V. García, J. S. Sánchez, R. Martín-Félez, and R. A. Mollineda, "Surrounding neighborhood-based SMOTE for learning from imbalanced data sets," *Prog. Artif. Intell.*, vol. 1, no. 4, pp. 347–362, Dec. 2012.

[33] C. Bellinger, S. Sharma, O. R. Zaıane, and N. Japkowicz, "Sampling a longer life: Binary versus one-class classification revisited," in *Proc. 1st Int. Workshop Learn. Imbalanced Domains, Theory Appl.*, 2017, pp. 64–78.

[34] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intell. Data Anal.*, vol. 6, no. 5, pp. 429–449, Oct. 2002.

[35] V. García, J. Sánchez, and R. Mollineda, "An empirical study of the behavior of classifiers on imbalanced and overlapped data sets," in *Proc. Iberoamerican Congr. Pattern Recognit.* Berlin, Germany: Springer, 2007, pp. 397–406.

[36] S.-W. Kim and B. J. Oommen, "On using prototype reduction schemes to enhance the computation of volume-based inter-class overlap measures," *Pattern Recognit.*, vol. 42, no. 11, pp. 2695–2704, Nov. 2009.

[37] H. Lu, Y. Meng, K. Yan, and Z. Gao, "Kernel principal component analysis combining rotation forest method for linearly inseparable data," *Cognit. Syst. Res.*, vol. 53, pp. 111–122, Jan. 2019.

[38] I. Tomek, "Two modifications of CNN," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, no. 11, pp. 769–772, Nov. 1976.

[39] F. Provost and T. Fawcett, "Robust classification for imprecise environments," *Mach. Learn.*, vol. 42, no. 3, pp. 203–231, 2001.

[40] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.

[41] R. Shatnawi, "The application of ROC analysis in threshold identification, data imbalance and metrics selection for software fault prediction," *Innov. Syst. Softw. Eng.*, vol. 13, nos. 2–3, pp. 201–217, Sep. 2017.

[42] A. Sun, E.-P. Lim, and Y. Liu, "On strategies for imbalanced text classification using SVM: A comparative study," *Decis. Support Syst.*, vol. 48, no. 1, pp. 191–201, Dec. 2009.

[43] L. A. T. Nguyen, O. Hirose, X. T. Dang, T. K. T. Le, T. Saethang, V. A. Tran, M. Kubo, Y. Yamada, and K. Satou, "Improving the prediction of protein-protein interaction sites using a novel over-sampling approach and predicted shape strings," *Annu. Rev. Res. Biol.*, vol. 3, no. 2, pp. 92–106, 2013.

[44] S. T. Jishan, R. I. Rashu, N. Haque, and R. M. Rahman, "Improving accuracy of students' final grade prediction model using optimal equal width binning and synthetic minority over-sampling technique," *Decis. Anal.*, vol. 2, no. 1, p. 1, 2015.

[45] A. Amin, S. Anwar, A. Adnan, M. Nawaz, N. Howard, J. Qadir, A. Hawalah, and A. Hussain, "Comparing oversampling techniques to handle the class imbalance problem: A customer churn prediction case study," *IEEE Access*, vol. 4, pp. 7940–7957, 2016.

[46] A. Luque, A. Carrasco, A. Martín, and A. de las Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix," *Pattern Recognit.*, vol. 91, pp. 216–231, Jul. 2019.

[47] J. Sayyad Shirabad and T. Menzies, "The PROMISE repository of software engineering databases," School Inf. Technol. Eng., Univ. Ottawa, Ottawa, ON, Canada, 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository

[48] A. Fernández, S. García, M. J. del Jesus, and F. Herrera, "A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets," *Fuzzy Sets Syst.*, vol. 159, no. 18, pp. 2378–2398, Sep. 2008.

[49] A. Fernández, M. J. del Jesus, and F. Herrera, "Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets," *Int. J. Approx. Reasoning*, vol. 50, no. 3, pp. 561–577, Mar. 2009.

[50] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[51] M. Tomczak and E. Tomczak, "The need to report effect size estimates revisited. An overview of some recommended measures of effect size," *Trends Sport Sci.*, vol. 1, no. 21, pp. 19–25, 2014.

[52] E. Drebitz, L.-P. Rausch, and A. K. Kreiter, "A novel approach for removing micro-stimulation artifacts and reconstruction of broad-band neuronal signals," *J. Neurosci. Methods*, vol. 332, Feb. 2020, Art. no. 108549.

[53] Y. Yang, J. Zhang, J. Carbonell, and C. Jin, "Topic-conditioned novelty detection," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2002, pp. 688–693.

[54] M. Markou and S. Singh, "Novelty detection: A review—Part 1: Statistical approaches," *Signal Process.*, vol. 83, no. 12, pp. 2481–2497, 2003.

[55] M.-R. Bouguelia, S. Nowaczyk, and A. H. Payberah, "An adaptive algorithm for anomaly and novelty detection in evolving data streams," *Data Mining Knowl. Discovery*, vol. 32, no. 6, pp. 1597–1633, Nov. 2018.

[56] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli, "Adversarially learned one-class classifier for novelty detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3379–3388.

[57] A. B. Hassanat, "Dimensionality invariant similarity measure," *J. Amer. Sci.*, vol. 10, no. 8, pp. 221–226, 2014.

[58] H. A. Abu Alfeilat, A. B. A. Hassanat, O. Lasassmeh, A. S. Tarawneh, M. B. Alhasanat, H. S. Eyal Salman, and V. B. S. Prasath, "Effects of distance measure choice on K-Nearest neighbor classifier performance: A review," *Big Data*, vol. 7, no. 4, pp. 221–248, Dec. 2019.

AHMAD S. TARAWNEH was born in Karak, Jordan. He received the B.S. and M.S. degrees in computer science from Mutah University, in 2013 and 2015, respectively. He is currently pursuing the Ph.D. degree in computer science with the Eötvös Loránd University (ELTE), Hungary, Budapest. He is currently working on the project of EFOP (image and video processing), which is sponsored by Hungarian government and co-financed by the European Social Fund. His main research interests include computer vision, deep learning, machine learning, and data mining.

AHMAD B. A. HASSANAT (Member, IEEE) was born and grew up in Jordan. He received the B.S. degree in computer science from Mutah University, Jordan, in 1995, the M.S. degree in computer science from Al al-Bayt University, Jordan, in 2004, and the Ph.D. degree in computer science from the University of Buckingham, U.K., in 2010. He has been a Faculty Member of the Department of Information Technology, Mutah University, since 2010. He is currently doing a sabbatical with the University of Tabuk, Saudi Arabia. His main research interests include computer vision, machine learning, big data, and pattern recognition.

KHALID ALMOHAMMADI received the bachelor's degree in computer science from Taibah University, Saudi Arabia, the M.Sc. degree (Hons.) in computer science from Newcastle University, U.K., in 2011, and the Ph.D. degree in computer science from Essex University, U.K. He then pursued a profession in teaching at Tabuk University. Building on this experience, he was awarded a scholarship to complete the master's degree in computer science (M.Sc.) at Newcastle University, which he obtained with distinction. His Ph.D. degree enables him to incorporate all his previous knowledge and experience in education and e-learning. The focus of his research is the development of theoretical and practical environments to enhance students' learning experience and engagement by using fuzzy logic systems. He currently works as an Assistant Professor with the Department of Computer Science, Community College, University of Tabuk.

DMITRY CHETVERIKOV received the Ph.D. and D.Sc. degrees, in 1988 and 2004, respectively. He is currently a Professor with the Eötvös Loránd University (ELTE) and a Scientific Adviser with the Institute for Computer Science and Control (SZTAKI), Budapest, Hungary. He is the author or coauthor of over 200 scientific articles (Google scholar H-index: 30). In general, he works in computer vision, 3D sensing, and sensor fusion. He took part in numerous research and application projects including those related to autonomous systems and vehicles. Since 2003, he has been a regular reviewer of proposals for EU FP6, FP7, and H2020. From 2008 to 2012, he was a member of the ERC Advanced Grants Committee PE6 (informatics and computer science).

COLIN BELLINGER received the Ph.D. degree from the University of Ottawa, in 2016. He is a Research Officer in data science for complex systems with the National Research Council of Canada. He held a Postdoctoral Fellowship with the Alberta Machine Intelligence Institute, University of Alberta, Edmonton, Canada, and the Donald Hill Fellowship in computer science with Dalhousie University, Halifax, Canada. He has coauthored more than 20 articles in international conferences and journals focused on machine learning, data mining, and health and computational security, and received two best paper awards. He was the Program Chair of the Graduate Students' AI Symposium, in 2017 and 2019. He has participated on numerous program committees and reviewed over ten international journals.