Soft Computing (2019) 23:10755–10767
https://doi.org/10.1007/s00500-018-3629-4

METHODOLOGIES AND APPLICATION

TLUSBoost algorithm: a boosting solution for class imbalance problem

Sujit Kumar[1] · **Saroj Kr. Biswas**[1] · **Debashree Devi**[1]

Published online: 20 November 2018
© Springer-Verlag GmbH Germany, part of Springer Nature 2018

Abstract
It is habitually assumed that the training sets used for learning are balanced. However, this hypothesis is not always true in real-world applications, and hence, there is a tendency of relying on the classification models that are biased towards the overrepresented class as traditional data mining algorithms are generally inclined towards building of suboptimal classification models. This class imbalance problem is common to many application domains such as data mining, machine learning, pattern recognition, etc. Several techniques have been proposed to alleviate the problem of class imbalance. RUSBoost is one of the ensemble learning approaches that uses random undersampling (RUS) for data resampling and AdaBoost technique for boosting, as a solution to class imbalance. However, RUS may cause the loss of significant information of dataset. Therefore, this paper proposes Tomek-link undersampling-based boosting (TLUSBoost) algorithm which uses Tomek-linked and redundancy-based undersampling (TLRUS) for data resampling and AdaBoost technique for boosting. TLRUS meticulously finds outliers using Tomek-link concept and then eliminates some of the probable redundant instances from the outliers. Hence, this algorithm reduces the loss of information and conserves the characteristics of the dataset, thereby helping the classifier to be trained appropriately. TLUSBoost method is validated with 16 benchmark datasets and compared with EasyEnsemble, BalanceCascade, SMOTEBoost and RUSBoost algorithms. Ten-fold cross-validation is applied to measure overall accuracy and F-measure metric of the models. Experimental results show that the proposed model is better than EasyEnsemble, BalanceCascade, SMOTEBoost and RUSBoost in both overall accuracy and F-measure performance metric.

Keywords Undersampling · Boosting · Data mining · Class imbalance problem · Tomek-link pair

1 Introduction

Classifier learning from class imbalanced datasets is a challenging problem in data mining community. The class imbalance problem occurs when the number of patterns that represent one class is much higher than the ones for the other classes. Its presence in many real-world applications has increasingly attracted the attention of researchers globally (Sun et al. 2009). The traditional data mining algorithms incline towards the class which is overrepresented (majority) as during the learning procedure, they ignore minority class patterns by treating them as noise, and hence, they might lose their classification ability in this scenario. Some of the domains prone to suffering from this kind of problem are fault diagnosis (Yang et al. 2009; Zhu and Song 2010), anomaly detection (Khreich et al. 2010; Tavallaee et al. 2010), medical diagnosis (Mazurowski et al. 2008), email foldering (Bermejo et al. 2011), face recognition (Liu and Chen 2005) or detection of oil spills (Kubat et al. 1998) etc.

Several techniques have been proposed to overcome this issue, and these techniques can be categorized into four groups based on how they deal with class imbalance problems. Algorithm level approach (also called internal) tries to adjust existing learning algorithms to bias the learning towards the minority class. This approach requires special knowledge of both the corresponding classifier and the application domain, which is not available in advance, and hence, this set of classifiers fails when the classes are not balanced. Data level (or external) approach rebalances the class

Communicated by V. Loia.

✉ Saroj Kr. Biswas
bissarojkum@yahoo.com

Sujit Kumar
kumar.sujit474@gmail.com

Debashree Devi
debashree.0748@gmail.com

[1] Department of Computer Science and Engineering, NIT Silchar, Silchar, Assam, India

distribution either by adding patterns into minority class, this process is known as oversampling or removing patterns from majority class, known as undersampling. This approach avoids the modification of the learning algorithm by trying to decrease the effect caused by data imbalance with a pre-processing step. Therefore, data level approach is not dependent on any classifier used and is usually more versatile. However, the drawback associated with oversampling is over fitting of the classifier due to elongated training time duration as duplicate instances are added into minority class. The drawback associated with undersampling is significant information loss of dataset as instances are removed from majority class (Seiffert et al. 2010; Galar et al. 2012). Cost-sensitive learning approach considers costs, such as misclassification costs, into account during classifier training and produces a classifier with lowest cost. In some applications, this approach is very important. This can be explained with an example from medical science domain, wherein if a cancer patient is wrongly classified as cancer-free patient then misclassification cost will be very high as it may lead to death of the patient. Several cost-sensitive learning techniques have been proposed to overcome class imbalance problem (Galar et al. 2012). The drawback associated with cost-sensitive learning is the need to define misclassification cost, which is usually not given in dataset (Elkan 2001).

Apart from these three categories, ensemble learning is another approach to overcome class imbalance problem by combining ensemble learning algorithm with one of the above mentioned approaches, particularly data resampling or cost-sensitive approach. Boosting, bagging and wagging are the most popular ensemble learning algorithms. While creating ensembles, creating diverse classifiers is a key factor to make them accurate. Recently, ensemble learning approaches which combine ensemble learning algorithm and data resampling technique have gained popularity to solve class imbalance problem recently such as IIVotes (Błaszczynski et al. 2010), SMOTEBoost (Chawla et al. 2002), RUSBoost (Seiffert et al. 2010), EasyEnsemble (Liu et al. 2009), BalanceCascade (Liu et al. 2009) or SMOTEBagging (Wang and Yao 2009); as ensemble learning algorithms, when directly applied to imbalanced datasets, cannot effectively solve the problem which underlies in the base classifiers themselves. The modification of the ensemble learning algorithm usually includes data level approaches to preprocess the data before training each classifier (Galar et al. 2012; Elkan 2001; Błaszczynski et al. 2010; Chawla et al. 2002; Liu et al. 2009; Wang and Yao 2009; Li et al. 2016; Biswas et al. 2017). In general, algorithm level and cost-sensitive approaches are more dependent on the problem, whereas data level and ensemble learning methods are more versatile since they can be used independently from the base classifier.

RUSBoost (Seiffert et al. 2010) is one of the most popular boosting techniques which combines random undersampling (RUS) and AdaBoost ensemble learning techniques to overcome class imbalance problem. However, RUS randomly removes patterns from majority class, which can cause significant loss of information from majority class. To reduce loss of information from training set during undersampling, Devi et al. (2017) have proposed a Tomek-linked and redundancy-based undersampling (TLRUS) technique where only outliers and noisy (Tomek-linked) patterns with higher redundancy and least contribution factor are subjected to elimination while performing undersampling. This concept is integrated with AdaBoost instead of RUS in this paper to overcome class imbalance problem. Hence, this paper proposes a Tomek-link undersampling-based boosting (TLUSBoost) algorithm which uses Tomek-linked and redundancy-based undersampling (TLRUS) for data resampling and AdaBoost technique for boosting. TLRUS meticulously finds outliers using Tomek-link concept, and then, some of the probable redundant instances are eliminated from the outliers. Hence, this algorithm decreases the loss of information and conserves the characteristics of the dataset, in turn helping the classifier to be trained appropriately. AdaBoost algorithm is trained with different weak learners: SVM, decision tree, K-NN and logistic regression.

The rest of the paper is organized as follows. Section 2 discusses some basic mechanisms, background studies related to the proposed method and some of the related works. Section 3 describes the research methodology of the proposed method. Results and their analysis are discussed in Sect. 4, and some conclusions are drawn in Sect. 5.

2 Literature survey

2.1 Undersampling

The process of removing patterns from majority class is called undersampling. The most common and basic undersampling technique is RUS which continuously removes patterns from majority class unless desired balance ratio between majority and minority class is achieved (Devi et al. 2017). The performance of RUS may be at par when closely similar patterns exist in majority class, but it causes the loss of significant information from majority class.

2.1.1 Tomek-Link pair

A pair of patterns is called a Tomek link if they belong to different classes and are each other's nearest neighbours (Devi et al. 2017). Undersampling can be done by removing all Tomek-link patterns from the majority class of a dataset.

2.1.2 Distance similarity measure

There are three popular similarity measure techniques: city-block distance (CD), Euclidean distance (ED) and cosine similarity (CS), which are used to find similarity between patterns. The definitions of these similarity measures for two patterns, x_i and x_j, comprising of m attributes are as follows:

i. *City-block distance (d_1)*

$$d_1 = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots\cdots |x_{im} - x_{jm}| \tag{1}$$

$$\text{and } sim(x_i, x_j) = 1/d_1 \tag{2}$$

ii. *Euclidean distance (d_2)*

$$d_2 = \sqrt{\left\{\left((x_{i1} - x_{j1})^2\right) + \left((x_{i2} - x_{j2})^2\right) + \cdots\cdots \left((x_{im} - x_{jm})^2\right)\right\}} \tag{3}$$

$$\text{and } sim(x_i, x_j) = 1/d_2 \tag{4}$$

iii. *Cosine similarity*: Cosine similarity provides a vector-based similarity measure that calculates the cosine of the angle between two patterns, as given below in Eq. (5) where the outcome lies within the range [0, 1]

$$sim(x_i, x_j) = \frac{\sum_{i,j}(x_i \times x_j)}{\sqrt{\left(\sum_i x_i^2\right)} \times \sqrt{\left(\sum_j x_j^2\right)}} \tag{5}$$

2.2 Redundancy-driven modified Tomek-link-based undersampling

It is one of the undersampling methods which is proposed to specifically alleviate the problem of random undersampling and to give a solution to class imbalance problem by Devi et al. (2017). This model finds outliers using Tomek-link pair and eliminates redundant patterns by measuring contribution factor. Only those Tomek-link patterns with higher redundancy and least contribution factor are subjected to elimination, while undersampling is performed on majority class.

2.3 AdaBoost

The basic idea behind the boosting is to create a highly accurate prediction rule by combining many relatively weak and inaccurate rules. AdaBoost (Freund and Schapir 1996; Cao et al. 2010) is one of the most commonly used and practical boosting algorithms. AdaBoost algorithm uses the whole training dataset to train each classifier serially, but gives more focus to the incorrectly classified patterns after each iteration for more accurate classification in the next iteration. The amount of focus is measured by assigning weight, which is initially equal for all patterns. The weights of misclassified patterns are increased, and the weights of correctly classified patterns are decreased after each iteration. Besides, an additional weight is assigned to each individual classifier (hypothesis) depending on its overall accuracy; figuratively assigning more confidence to more accurate classifier. Finally, when a new pattern is submitted, each classifier gives a weighted vote and the class label is determined by majority of voting.

2.4 Classifiers/weak learners

Different classifiers are trained as weak learners in the AdaBoost algorithm to predict the target class of new patterns. Some of the popular classifiers which are used as weak learners in TLUSBoost are as follows:

2.4.1 Support vector machine (SVM)

SVM is a widely used technique for data classification and regression, which is commonly used as classifier due to its theoretical foundation and superiority to many other classifiers. SVM (Fradkin and Muchnik 2000; Gunn 1998) is a classifier which maps patterns into a higher-dimensional plane using either linear or nonlinear kernel functions. It uses a linear separating hyper-plane to maximize the distance between two classes and hence is also known as "hyper-plane classifier". In binary classification problem, the two groups are separated by a higher-dimensional hyper-plane in accordance with structural risk minimization principle. The objective is to find a linear separating hyper-plane given below:

$$wf(x) + b = 0 \tag{6}$$

where $f(x)$ is a nonlinear function into a higher-dimensional feature space, w is weight vector, and b is bias.

2.4.2 Decision tree (DT)

Decision tree classifier is a powerful and popular tool used for classification technique, which applies a straightforward idea to solve the classification problems. The decision tree works by recursively selecting the best attribute to split the training data and expanding the leaf nodes of the tree until the stopping criterion is met (Pallara 1992). Decision tree represents rules, which can be understood by humans and be used in knowledge systems such as database. There are some expected features to build a decision tree such as instances or cases must be expressible in terms of a fixed collection of properties or attributes. The target function has discrete

output values. Enough training instances should be provided for learning of the model.

2.4.3 K-nearest neighbour (K-NN)

Perhaps the most straightforward classifier in machine learning is the K-nearest neighbour (K-NN) classifier. The classification is done by identifying the K-nearest neighbours to a new pattern (Sutton 2012). Nearest neighbours are found by calculating distance similarity measure such as city-block distance, Euclidean distance or cosine distance. The main steps of this classifier can be summarized as follows (Sutton 2012):

- A positive integer, K is specified.
- K-nearest neighbours of the pattern under consideration for classification are selected from the concerned dataset.
- The most common classification of these nearest neighbours is determined.
- This classification label is assigned to the new pattern.

2.4.4 Logistic regression (LR)

Logistic regression analyses the relationship between multiple independent variables and a categorical dependent variable and estimates the probability of occurrence of an event by fitting data to a logistic curve (Park 2013). There are two models of logistic regression: binary logistic regression and multinomial logistic regression. Binary logistic regression models the probability of occurrence of one of the two classes on dichotomous criterion and is used when the dependent variables are dichotomous and the independent variables are either continuous or categorical. In case the dependent variables have more than two categories, multinomial logistic regression is used. Detailed description of LR can be found in Park (2013).

2.5 Related work

Class imbalance problem has gained significant attention from data mining and machine learning researchers' community. Many techniques have been proposed to overcome class imbalance problem. Phung et al. (2009) reviewed the existing approaches for solving the class imbalance approach and also proposed a new approach to deal with class imbalance problem using supervised and unsupervised learning. Galar et al. (2012) presented a survey of the existing methods to overcome class imbalance problem and concluded that ensemble-based algorithms are valuable for improving the results that are obtained by the usage of data pre-processing techniques and training a single classifier. They also observed that ensemble learning approaches such as RUSBoost or UnderBagging perform better than many other conventional ensemble learning approaches. Chawla et al. (2002) proposed SMOTEBoost, a hybrid method to alleviate class imbalance problem which combines SMOTE (Synthetic Minority Oversampling) as data pre-processing with AdaBoost as ensemble learning algorithm. Seiffert et al. (2010) proposed an alternative to SMOTEBoost by combining RUS and AdaBoost, which is simple and faster to build model compared to SMOTEBoost. Mustafa et al. (2015) proposed a method which combines RUS with MultiBoost ensemble learning algorithm. Liu et al. (2009) proposed EasyEnsemble and BalanceCascade methods to overcome class imbalance problem. These methods are specifically designed to minimize the information loss during undersampling.

3 Proposed model

The framework of the method is shown in Fig. 1. A dataset is divided into majority class patterns and minority class patterns. Pre-processing on majority class patterns is performed for undersampling using Tomek-link and redundancy-based undersampling method (Devi et al. 2017). The pre-processing part consists of three steps: (i) elimination of outliers from majority class, (ii) detection of redundant pairs and (iii) elimination of redundant patterns. Now, these refined majority class patterns are combined with minority class patterns to form a new training set. New refined training set and original training set both are then fed to the AdaBoost algorithm for learning. Here, refined training set is used for training of weak learner and original training dataset is used for prediction, to calculate the pseudo-loss. The proposed method is broadly divided into two phases: (a) *pre-processing* and (b) *boosting*.

3.1 Pre-processing

It is very common to have class imbalance problem in binary classification. Hence, the main objective of pre-processing is to balance the uneven distribution of minority and majority patterns. Pre-processing on majority class patterns is performed using Tomek-link and redundancy-based undersampling method (TLRUS) (Devi et al. 2017). Steps for pre-processing are given below.

3.1.1 Elimination of outliers from majority class

In this step, outliers are determined and eliminated based on Tomek-link pair and a threshold value (r) provided accordingly. Let S be a dataset with n patterns and m be the number of attributes in each pattern. Let each pattern is characterized as $\left(\overrightarrow{x_a}, y_b\right)$ where $\overrightarrow{x_a}$ is the input vector for a^{th}- pattern, denoted as $\overrightarrow{x_a} = (x_{a1}, x_{a2}, \ldots, x_{am})$ such that m is the num-

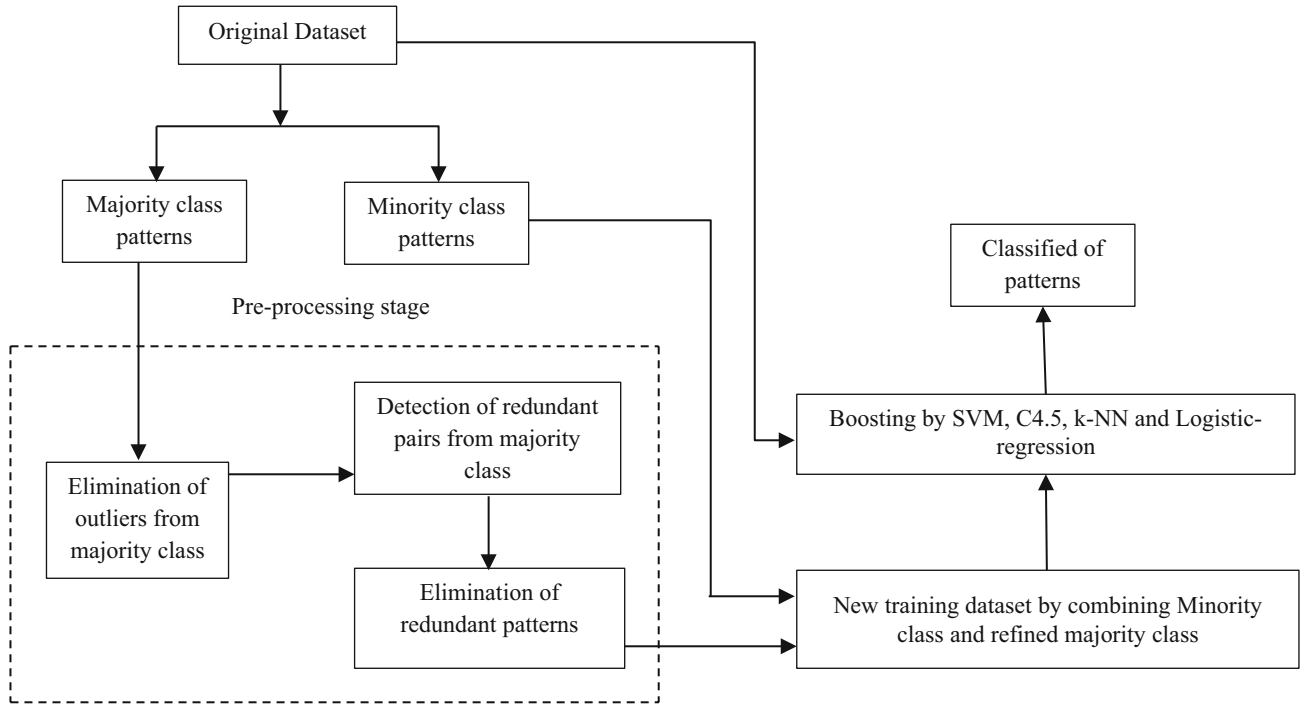


Fig. 1 Schematic diagram of the TLUSBoost algorithm

ber of attributes and $1 \leq a \leq n$, and $y_b = (y_1, y_2, \ldots y_c)$ is the designated class label out of total c-classes. The procedure to eliminate outliers from majority class is as follows.

Notation:

$S_{n\times m}$: An imbalanced dataset, with n *patterns* and m attributes.

S_{min}: Set of minority class patterns, S_{min} : $\{x_i | i = 1, 2, \ldots., p\}$, where p is the number of minority class patterns.

S_{maj}: Set of majority class patterns, S_{maj} : $\left\{x_j | j = 1, 2, \ldots, q\right\}$, where q is the number of majority class patterns.

TL: Set of Tomek-link patterns.

$minor_{x_j}$: The number of minority patterns, to which majority pattern, x_j has associated as Tomek-link pair

$TL^{'}$: Revised subset of TL, after elimination of majority patterns determined as outliers.

$S^{'}_{maj}$: Set of majority class patterns after removing outliers.

Input: S_{min}, S_{maj}, threshold (r) to determine majority patterns as outliers. Set nearest neighbour $k=1$.

Output: Reduced majority subset, $S^{'}_{maj}$

Begin:

i. Divide the entire dataset, $S_{n\times m}$ into two subsets, S_{min} and S_{maj}.
ii. For each $x_i \in S_{min}$, determine its K-nearest neighbour from x_j such that $x_j \in S_{maj}$ by using Euclidean distance.
iii. The nearest neighbour pairs $\left(x_i, x_j\right)$ are stored in subset, TL as

$$TL = \left\{\left(x_i, x_j\right) | x_i \in S_{min}, x_j \in S_{maj}\right\}.$$

iv. For each $x_j \in S_{maj}$, determine $minor_{x_j}$.
v. If $minor_{x_j} > t$, x_j is considered to be "outlier", lying in the minority region, and stored in subset, OUT as

$$OUT = \left\{x_j | \forall x_j, minor_{x_j} > t\right\}$$

vi. Subset, TL is updated to $TL^{'}$ as $TL^{'} = TL - \{OUT\}$
vii. Subset, S_{maj} is updated to $S^{'}_{maj}$ as $S^{'}_{maj} = S_{maj} - \{OUT\}$

Patterns in the set OUT are termed as outliers and are removed. $TL^{'}$ comprises of remaining Tomek-link paired patterns.

3.1.2 Detection of redundant pairs

Redundant pairs are detected from the remaining Tomek-link paired patterns. Redundancy and noise factors are considered together to detect majority class patterns residing in the proximity of the decision boundary and having high similarity with other majority patterns. The similarity measures considered are ED, CB and CS as discussed in subsection 2.1.2. The procedure to detect redundant pairs from the remaining Tomek-link paired patterns is as follows.

Notation:

S'_{maj}: Updated set of majority class patterns
RD: Set of most redundant majority pair patterns for each $x_m, x_m \in S'_{maj}$.
TL_RD: Intersection of sets TL' and RD.

Procedure
Input:

S'_{maj}, TL', k = number of most redundant patterns used to create, RD. Set k=1.

Begin:

i. For each $x_m \in S'_{maj}$, determines its redundant pairs from S'_{maj} by using the ED, CB and CS distance similarity measures, defined in Eqs. (1)–(3) and is stored in the subset, RD as

$$RD = \left\{(x_m, x_r) | \forall x_{m,} x_r \in S'_{maj} \; and \; sim(x_m, x_r) = maximum\right\}$$

ii. A revised subset, TL_RD, is created by the intersection of TL' and RD, as

$$TL_RD = \left\{\cap x_m \in S'_{maj} | x_m \in TL', x_m \in RD\right\}$$

Redundant majority patterns which are in set TL_RD are considered for next step.

3.1.3 Elimination of redundant patterns

To eliminate a redundant majority pattern from a majority redundant pair (x_u, x_v) where $x_u, x_v \in TL_RD$, a contribution factor, $Contr_x$ is employed, which is determined as follows:

$$Contr_x = \frac{1}{N} \times \left\{\left(\sum_{a=1}^{n}\sum_{k=1}^{m} \ln f(x_{ak}|C_l)\right)\right\} \quad (7)$$

where $P(C_l)$ denotes the prior probability of class, C_l and $\ln f(x_{ak}|C_l)$ is the log-likelihood function. Hence, majority patterns reclined to the set TL_RD are evaluated in terms of their redundancy and contribution factors. Patterns with higher redundancy and least contribution are selected for elimination. Redundant eliminated patterns are stored in set TL_RD' by the following criteria:

$$TL_RD' = max\left(\frac{sim(x_u, x_v)}{Contr_{x_u}}, \frac{sim(x_u, x_v)}{contr_{x_v}}\right); x_u, x_v \in TL_RD' \quad (8)$$

If two majority patterns are found to be redundant, the one having lesser contribution factor will be discarded.

Now, the refined dataset S_R is obtained by the following steps:

i. Update S'_{maj} to S''_{maj} as $S''_{maj} = S'_{maj} - TL_RD'$
ii. Obtain the refined training set with balanced distribution as $S_B = S''_{maj} \cup S_{min}$.

3.2 Boosting

Boosting is implemented to improve classifier performance. AdaBoost boosting algorithm (Freund and Schapir 1996; Cao et al. 2010) iteratively builds an ensemble of different learning models and is applied here. During the training, patterns weights are modified with an aim to correctly classify patterns in the next iteration, which were incorrectly classified during the current iteration. The proposed TLUSBoost technique is mainly performed on the refined dataset, S_B as well as on the original dataset, S. The procedure of TLUSBoost is given below.

Procedure:
Notations:

T = number of classifiers in the ensemble.
t = an iteration between one and the maximum number of iterations, T.
h^t = the weak hypothesis on iteration t, trained using some classification algorithm (*Weak Learner).*
$h^t_{(x_a)}$ = the output of hypothesis, h^t *for pattern,*x_a.
D^t_a = the weight of the ath pattern for iteration, t.

Begin:

i. Initialize $D^1_a = \frac{1}{n}$, $\forall\, a$.
ii. Do for t= *1, 2, ……, T.*

a. Create training dataset S_B using TLRUS algorithm, with new weight distribution, $D^{t\prime}$.
b. Call one of the *Weak Learners*, providing it with patterns of S_B and their weight $D^{t\prime}$.
c. Obtain a hypothesis h^t by learning S_B.

d. Calculate the pseudo-loss for S and $D^{t\prime}$:

$$e^t = \sum_{\substack{a=1,n \\ ((x_a,y)|y_a \neq y)}} e^t_{(x_a)} \times D^{t'}_{(x_a)}$$

e. Calculate the weight update parameter:

$$\alpha^t = \log\left(\frac{1-e^t}{e^t}\right)$$

$$D^{(t+1)}_{x_a} = D^{t'}_{(x_a)} exp\left(\alpha^t \cdot e^t_{(x_a)}\right)$$

g. Re-normalize $D^{(t+1)}_{x_a}$ such as $\sum_{a=1}^{n} D^{(t+1)}_{x_a}$

iii. Output the final hypothesis, for each test pattern, x_j as

$$H(x_j) = argmax \sum_{t=1}^{T} \alpha^t \cdot h^t_{(x_j)}$$

The weights of each pattern are initialized to $\frac{1}{n}$. In step (ii), T weak learners are iteratively trained. TLRUS technique is used to undersample majority class in order to obtain the balanced training dataset, S_B. As a result, S_B will have a new weight distribution, D'_t. For each iteration, t ($1<t<T$), a *Weak Learner* creates the hypothesis, h^t. The final hypothesis $H(x_j)$ is returned as a weighted vote of the T-hypothesis, for each test pattern, x_j.

4 Results and discussion

The proposed model is validated with 16 benchmark datasets taken from the University of California, Irvine (UCI) machine learning repository (https://archive.ics.uci.edu/ml. 10th November 2016). The statistics of the datasets are presented in Table 1. All the datasets are of binary class. The imbalance ratio (IR) is determined as $\frac{S_{maj}}{S_{min}}$.

The proposed TLUSBoost is compared with EasyEnsemble (Liu et al. 2009), BalanceCascade (Liu et al. 2009), SMOTEBoost (Chawla et al. 2002) and RUSBoost (Seiffert et al. 2010) methods. To evaluate the performances of all the methods, overall accuracy and F-measure are used as performance measures. A method having higher overall accuracy and higher F-measure is considered as a better performer than those with lower values. Ten-fold cross-validation is applied for performance evaluation.

Tables 2, 3, 4 and 5 show overall accuracy of EasyEnsemble, BalanceCascade, SMOTEBoost, RUSBoost and TLUSBoost methods in different datasets with decision tree, K-NN, logistic regression and SVM classifiers, respectively. The proposed TLUSBoost gives optimal performance for different datasets for different thresholds. In the proposed method, threshold value is experimental and plays significant role in detecting redundant outliers. It is observed from the results shown in Table 2 that TLUSBoost produces better overall accuracy than all other methods for 8 out of 16 datasets, SMOTEBoost produces for 5 out of 16 datasets, EasyEnsemble produces for 2 out of 16 datasets, and BalanceCascade produces for 1 out of 16 datasets. If the performance of TLUSBoost is individually compared with each of baseline methods, the following observations are made

i. TLUSBoost produces better overall accuracy than EasyEnsemble for 12 datasets.
ii. TLUSBoost produces better overall accuracy than BalanceCascade for 13 datasets.
iii. TLUSBoost produces better overall accuracy than SMOTEBoost for 10 datasets.
iv. TLUSBoost produces better overall accuracy than RUSBoost for 13 datasets.

Hence, it can be concluded that the overall accuracy obtained by TLUSBoost method with decision tree is far better than EasyEnsemble, BalanceCascade, RUSBoost and SMOTEBoost.

Results given in Table 3 show that TLUSBoost produces better overall accuracy than EasyEnsemble, BalanceCascade, RUSBoost and SMOTEBoost for 8 datasets, while SMOTEBoost produces better overall accuracy than TLUSBoost, RUSBoost, EasyEnsemble and BalanceCascade for 4 datasets. RUSBoost produces better overall accuracy for 2 datasets, and EasyEnsemble also produces better overall accuracy for 2 datasets. When the performance of TLUSBoost is individually compared with each of the other methods, it is clear that the TLUSBoost is better than the other methods. Therefore, it is established that overall accuracy obtained by TLUSBoost method with K-NN is better than EasyEnsemble, BalanceCascade, RUSBoost and SMOTEBoost.

Similarly, results given in Table 4 show that TLUSBoost produces better overall accuracy than other methods for 10 datasets, while SMOTEBoost produces better overall accuracy than others for 2 datasets, RUSBoost produces better overall accuracy than others for 3 datasets and BalanceCascade for 1 dataset. When the performance of TLUSBoost is individually compared with each of the other methods, then TLUSBoost performs far better than the other methods. Therefore, it can be concluded that TLUSBoost method with logistic regression is better that other baseline methods.

Similarly, it can be observed from results given in Table 5 that TLUSBoost produces better overall accuracy than other

Table 1 Statistics of datasets

Sl. no	Dataset	Instances/patterns	Attributes	No of minority	No. of majority	Imbalance ratio (IR)*
1	Blood transfusion	748	5	178	570	3.2022
2	PC1	1109	22	77	1032	13.4025
3	CM1	498	22	49	449	9.1632
4	Mammographic	1035	6	445	590	1.3258
5	Wisconsin	683	11	106	590	5.5660
6	Yeast	1005	9	99	906	9.1515
7	Led	798	8	37	761	20.5675
8	Glass	897	10	60	837	13.95
9	Diabetes	768	9	268	506	1.8880
10	Segment	2308	20	329	1979	6.01519
11	Vehicle	846	19	217	846	3.8986
12	Vehicle2	846	19	199	846	4.2512
13	Pima Indian	768	9	268	500	1.8656
14	CMC	850	10	208	642	3.0865
15	Glass6	214	10	29	185	6.3793
16	Phoneme	5404	6	1586	3818	2.4073

Table 2 Performance of EasyEnsemble, BalanceCascade, SMOTEBoost, RUSBoost and TLUSBoost with decision tree

	Datasets	EasyEnsemble (avg. accuracy)	BalanceCascade (avg. accuracy)	SMOTEBoost (avg. accuracy)	RUSBoost (avg. accuracy)	TLUSBoost (avg. accuracy)	Experimental threshold value
1	Blood transfusion	66.4665	62.67	62.3416	65.5452	74.2082	4
2	PC1	76.02	77.38	90.7744	72.4222	90.6153	4
3	CM1	72.00	60.00	86.7667	69.7083	88.175	2
4	Mammographic	80.73	78.13	77.7914	78.411	79.0608	13
5	Wisconsin	77.70	74.82	77.0476	76.342	80.342	6
6	Yeast	80.60	65.86	88.3601	83.4777	91.0386	4
7	LED	93.08	89.31	96.0494	89.7531	96.8921	3
8	Glass	95.53	84.92	71.5065	91.9732	97.6513	6,7
9	Diabetes	73.38	70.78	68.0208	73.05457	75.1359	3
10	Segment	99.78	99.57	99.6528	97.7918	95.1935	12
11	Vehicle	77.51	73.96	79.4365	72.47204	75.8954	2
12	Vehicle2	94.08	87.8	96.3475	93.3711	94.0944	**7**
13	Pima Indian	72.55	71.99	71.3593	73.1775	75.1359	3
14	CMC	52.94	58.80	65.9976	60.0061	62.8978	13
15	Glass6	90.70	95.35	77.2727	93.0114	95	11
16	Phoneme	86.77	87.97	89.5986	84.8423	84.4771	8

methods for 8 datasets, while SMOTEBoost produces better overall accuracy than others for 3 datasets, RUSBoost produces better overall accuracy than others for 4 datasets, and EasyEnsemble for 1 dataset. When the performance of TLUSBoost is individually compared with each of the other methods, then TLUSBoost performs far better than the other methods. In conclusion, TLUSBoost method with SVM is better than the other baseline methods.

It is clear from the experimental results that the proposed TLUSBoost model is better than RUSBoost, SMOTEBoost, BalanceCascade and EasyEnsemble for all the datasets and for all the classifier platforms. The proposed model produces better accuracy as it finds and eliminates the most redundant

Table 3 Performance of EasyEnsemble, BalanceCascade, SMOTEBoost, RUSBoost and TLUSBoost with K-NN

	Datasets	EasyEnsemble (avg. accuracy)	BalanceCascade (avg. accuracy)	SMOTEBoost (avg. accuracy)	RUSBoost (avg. accuracy)	TLUSBoost (avg. accuracy)	Experimental threshold value
1	Blood transfusion	33.0886	63.8664	57.2749	64.7416	69.1452	3
2	PC1	59.2618	58.4795	86.9157	78.7615	90.9627	5
3	CM1	61.8667	61.8667	82.1333	75.35	84.3583	3–15
4	Mammographic	59.0833	60.2853	73.7821	77.33	77.1619	11
5	Wisconsin	71.3420	71.6277	77.6104	78.1991	80.1991	1
6	Yeast	54.5421	55.1207	85.876	87.0627	91.1531	2
7	LED	41.2560	95.4106	95.1185	88.9909	96.7687	2
8	Glass	79.5651	80.6251	97.7548	80.4636	93.9693	8
9	Diabetes	97.0085	71.8788	68.2113	74.1017	70.1766	4
10	Segment	76.1307	96.9648	98.5688	97.1805	99.0036	8,9
11	Vehicle	76.1307	70.0901	72.5781	72.47204	73.3958	4
12	Vehicle2	90.9663	89.4829	93.3603	89.3537	93.4946	5
13	Pima Indian	72.1455	71.8788	69.4078	73.1706	70.1801	13,14
14	CMC	48.5282	50.9942	62.6255	54.3482	59.2166	3
15	Glass6	87.2727	86.8182	94.4886	90.5114	92.2159	2
16	Phoneme	85.1383	84.5762	88.7111	86.0458	85.1387	11

Table 4 Performance of EasyEnsemble, BalanceCascade, SMOTEBoost, RUSBoost and TLUSBoost with logistic regression

	Datasets	EasyEnsemble (avg. accuracy)	BalanceCascade (avg. accuracy)	SMOTEBoost (avg. accuracy)	RUSBoost (avg. accuracy)	TLUSBoost (avg. accuracy)	Experimental threshold value
1	Blood transfusion	38.4551	75.5647	64.785	71.4082	76.8749	2
2	PC1	65.9954	65.2954	80.3085	84.3167	91.4692	5
3	CM1	72.1583	72.1583	86.185	79.7417	88.175	3
4	Mammographic	72.6768	77.9999	78.58	79.5367	78.05082	11
5	Wisconsin	76.6277	75.9993	78.5022	79.2078	80.7965	1
6	Yeast	54.5060	56.1892	78.2312	87.7197	91.1273	9
7	LED	45.3301	95.4106	93.864	91.2131	96.6452	2
8	Glass	88.5523	89.2100	89.3218	89.4138	94.751	3
9	Diabetes	75.1394	75.1394	73.8095	75.4026	75.6485	4
10	Segment	89.5983	98.4404	77.1837	88.3862	88.3862	4
11	Vehicle	77.5599	77.5599	72.8711	78.825	79.801	6
12	Vehicle2	91.7473	95.6296	96.9213	93.3711	96.2237	5
13	Pima Indian	75.1394	74.5667	75.529	75.6623	75.6485	4
14	CMC	56.5361	56.5361	57.5949	61.5177	65.6732	3
15	Glass6	86.7443	87.5672	96.3471	92.5	94.2045	10
16	Phoneme	72.4090	72.2134	72.87	74.8706	73.926	9

Table 5 Performance of EasyEnsemble, BalanceCascade, SMOTEBoost, RUSBoost and TLUSBoost with SVM

	Datasets	EasyEnsemble (avg. accuracy)	BalanceCascade (avg. accuracy)	SMOTEBoost (avg. accuracy)	RUSBoost (avg. accuracy)	TLUSBoost (avg. accuracy)	Experimental threshold value
1	Blood transfusion	42.9954	75.9012	63.5489	77.1416	76.8789	2
2	PC1	76.8299	76.8299	84.7339	91.5488	90.9627	5
3	CM1	85.3667	85.3667	81.7667	87.5667	89.9667	1
4	Mammographic	85.3667	76.5623	81.7667	77.8351	77.5515	7
5	Wisconsin	77.0476	77.476	78.7706	77.0563	77.645	2
6	Yeast	63.9119	63.9119	86.4377	89.8247	90.5487	3
7	LED	38.5400	95.4106	95.556	91.2131	97.2625	2
8	Glass	90.3569	91.8502	86.6092	89.6973	93.3333	2
9	Diabetes	74.2303	74.2303	74.097	74.213	75.1359	5
10	Segment	97.0988	97.0988	99.4801	97.7918	99.61	8
11	Vehicle	71.0254	71.0254	76.8308	76.1423	77.5599	5
12	Vehicle2	92.5534	92.5534	96.9296	93.3711	94.6768	3
13	Pima Indian	74.2303	74.2303	75.529	75.7991	75.1359	5
14	CMC	53.1334	53.1334	57.5949	64.1065	67.4786	3
15	Glass6	88.1818	89.909	96.1932	93.4091	92.5	2
16	Phoneme	71.1133	72.3456	73.8319	76.9231	76.2764	6

instances from the training data by setting three criteria: (i) detection of outliers from majority class, (ii) detection of redundant pairs and (iii) elimination of redundant patterns. In the proposed model, some criteria are scientifically set to find and eliminate redundant instances, whereas RUSBoost randomly eliminates instances, and hence, RUSBoost may have significant information loss while performing undersampling.

The performance of the proposed model is illustrated in Tables 6, 7, 8 and 9 in terms of F-measure. The proposed model produces better F-measure performance for 4 datasets with decision tree, for 6 datasets with K-NN, for 6 datasets with logistic regression and for 7 datasets with SVM classifiers than all the other baseline methods. When the F-measure performance of TLUSBoost is individually compared with each of the other methods, then it can be observed that TLUSBoost performs far better than other methods. As an example, F-measure performance of TLUSBoost with decision tree is individually compared with each of the baseline methods and the following observations are made:

i. TLUSBoost produces better F-measure than EasyEnsemble for 9 datasets.
ii. TLUSBoost produces better F-measure than BalanceCascade for 8 datasets.
iii. TLUSBoost produces better F-measure than SMOTEBoost for 9 datasets.
iv. TLUSBoost produces better F-measure than RUSBoost for 11 datasets.

From all the experimental results, it has been established that the proposed model is superior to EasyEnsemble, BalanceCascade, RUSBoost and SMOTEBoost methods.

4.1 Discussion

The threshold values yielding optimal results are presented in the performance Tables 2, 3, 4 and 5.

Threshold value setting Threshold is used to detect an outlier from the set of Tomek-link paired instances during data pre-processing. The prime motivation is to detect more number of outliers among the Tomek-link instances. However, Devi et al. (2017) have suggested that a very small value of threshold is not realistic as it may lead to deletion of important majority class instances. Similarly, a very high threshold value can erroneously detect outliers as potential instances. A sensitivity analysis is performed to evaluate the impact of varying threshold values over the performance of the redundancy-driven modified Tomek-link-based undersampling. It is observed that threshold values in the range of 2–15 are adequate in context to problem definition. TLUSBoost selects the optimal threshold by verifying the performance of TLUSBoost for all the threshold values between 2 and 15. A threshold value is selected as optimal for which TLUSBoost gives maximum performance. It should be noted that optimal threshold values are different for different datasets.

Time complexity of TLUSBoost There are two dominant operations: data pre-processing by redundancy-driven modified Tomek-link-based undersampling and build hypothesis

Table 6 F-measures of EasyEnsemble, BalanceCascade, SMOTEBoost, RUSBoost and TLUSBoost with decision tree

	Datasets	EasyEnsemble	BalanceCascade	SMOTEBoost	RUSBoost	TLUSBoost
1	Blood transfusion	0.4848	0.4615	0.2118	0.3366	0.4083
2	PC1	0.3117	0.3243	0.3185	0.3371	0.3165
3	CM1	0.3636	0.2308	0.2098	0.2261	0.0806
4	Mammographic	0.8663	0.7823	0.7312	0.7539	0.772
5	Wisconsin	0.5231	0.5205	0.2656	0.4965	0.5409
6	Yeast	0.480	0.3704	0.4347	0.4834	0.49
7	LED	0.5600	0.4138	0.6483	0.5751	0.6174
8	Glass	0.7500	0.4255	0.8536	0.6305	0.8206
9	Diabetes	0.6612	0.6018	0.5689	0.6547	0.6927
10	Segment	0.9925	0.9851	0.9863	0.9273	0.6798
11	Vehicle	0.6667	0.6071	0.5705	0.5793	0.6078
12	Vehicle2	0.8864	0.8780	0.9116	0.8653	0.8745
13	Pima Indian	0.5214	0.5124	0.5863	0.6338	0.6927
14	CMC	0.3750	0.3478	0.2775	0.3934	0.3888
15	Glass6	0.7500	0.833	0.323	0.7748	0.7648
16	Phoneme	0.8028	0.8127	0.8207	0.7686	0.767

Table 7 F-measures of EasyEnsemble, BalanceCascade, SMOTEBoost, RUSBoost and TLUSBoost with K-NN

	Datasets	EasyEnsemble	BalanceCascade	SMOTEBoost	RUSBoost	TLUSBoost
1	Blood transfusion	0.2470	0.2514	0.2262	0.2529	0.2941
2	PC1	0.2269	0.2309	0.3306	0.3059	0.3326
3	CM1	0.2456	0.2456	0.2421	0.2401	0.1771
4	Mammographic	0.2107	0.3564	0.6677	0.73	0.754
5	Wisconsin	0.4669	0.4698	0.3555	0.541	0.5177
6	Yeast	0.2934	0.2494	0.4605	0.5198	0.564
7	LED	0.1834	0	0.6468	0.4924	0.6018
8	Glass	0.3542	0.3612	0.8537	0.3041	0.3713
9	Diabetes	0.6428	0.6396	0.8124	0.6109	0.6392
10	Segment	0.9038	0.9038	0.3807	0.9104	0.9645
11	Vehicle	0.5709	0.5849	0.9504	0.5793	0.5838
12	Vehicle2	0.8260	0.8164	0.8045	0.8138	0.8753
13	Pima Indian	0.6428	0.6214	0.5757	0.6302	0.6312
14	CMC	0.3697	0.3569	0.32	0.3467	0.3578
15	Glass6	0.7009	0.7148	0.8124	0.7329	0.7144
16	Phoneme	0.7747	0.7574	0.8045	0.7809	0.7743

using a weak learner. The asymptotic time complexity of data pre-processing by redundancy-driven modified Tomek-link-based undersampling is $O(nd)$ where d is the number of features and n is the number of patterns in training set, and the asymptotic time complexity to build a hypothesis using SVM (weak learner) is $O\left(max(n,d)min(n,d)^2\right)$ which can be written as $O\left(n^3\right)$ because usually the value of n is much higher than the value of d. However, in addition, there are some other independent operations involved for weight updating of patterns in AdaBoost which either take constant time or O(n)-time, but build hypothesis using weak learner is dominant. Since $O\left(n^3\right)$ is very large compared to $O(nd)$, the asymptotic time complexity of TLUSBoost is $O\left(n^3\right)$ when SVM is used as weak learner.

5 Conclusion

Class imbalance is a challenging problem in data mining and machine learning community. This paper proposes a

Table 8 F-measures of EasyEnsemble, BalanceCascade, SMOTEBoost, RUSBoost and TLUSBoost with logistic regression

	Datasets	EasyEnsemble	BalanceCascade	SMOTEBoost	RUSBoost	TLUSBoost
1	Blood transfusion	0.2903	0.3120	0.4026	0.3053	0.3302
2	PC1	0.2383	0.2383	0.2667	0.286	0.1592
3	CM1	0.2679	0.2679	0.2515	0.3739	0.3078
4	Mammographic	0.2679	0.4235	0.7634	0.7638	0.767
5	Wisconsin	0.4981	0.4981	0.5323	0.5418	0.4951
6	Yeast	0.2671	0.2330	0.4618	0.5197	0.5515
7	LED	0.1319	0	0.6098	0.532	0.5765
8	Glass	0.4125	0.4250	0.5421	0.4982	0.4263
9	Diabetes	0.6657	0.6657	0.6518	0.6558	0.6783
10	Segment	0.2853	0.9465	0.6296	0.9724	0.1864
11	Vehicle	0.6473	0.6373	0.61	0.648	0.6479
12	Vehicle2	0.7281	0.9117	0.92	0.8749	0.92
13	Pima Indian	0.6657	0.5875	0.6672	0.6535	0.6783
14	CMC	0.4363	0.4365	0.4296	0.4043	0.36
15	Glass6	0	0.7148	0.6657	0.7578	0.6524
16	Phoneme	0.6246	0.6246	0.62	0.6273	0.6322

Table 9 F-measures of EasyEnsemble, BalanceCascade, SMOTEBoost, RUSBoost and TLUSBoost with SVM

	Datasets	EasyEnsemble	BalanceCascade	SMOTEBoost	RUSBoost	TLUSBoost
1	Blood transfusion	0.1948	0.1012	0.3703	0.109	0.1061
2	PC1	0.2175	0.2165	0.195	0.281	0.3326
3	CM1	0.2766	0.2766	0.3641	0.1715	0.0333
4	Mammographic	0.2766	0.3456	0.76	0.7531	0.7622
5	Wisconsin	0.5241	0.5241	0.5155	0.5382	0.5056
6	Yeast	0.3301	0.3301	0.468	0.5828	0.4147
7	LED	0.1640	0	0.6272	0.5342	0.6758
8	Glass	0.3124	0.3340	0.3877	0.1492	0.2342
9	Diabetes	0.6589	0.6657	0.6499	0.6423	0.674
10	Segment	0.9062	0.9620	0.9808	0.9273	0.9849
11	Vehicle	0.5686	0.6473	0.6346	0.588	0.5734
12	Vehicle2	0.8642	0.8642	0.9356	0.8749	0.8972
13	Pima Indian	0.6589	0.4256	0.6672	0.6496	0.674
14	CMC	0.4299	0.4299	0.4296	0.3793	0.2246
15	Glass6	0.6772	0.7572	0.8348	0.8002	0.7502
16	Phoneme	0.6243	0.6173	0.6362	0.6405	0.6485

boosting model, TLUSBoost, to solve class imbalance problem for classification. Here, Tomek-link and redundancy-based undersampling (TLRUS) technique is integrated with AdaBoost. TLRUS finds and eliminates the most redundant instances from the training data by setting three criteria: (i) detection of outliers from majority class, (ii) detection of redundant pairs and (iii) elimination of redundant patterns, and then AdaBoost algorithm is trained with different learners, namely SVM, decision tree, K-NN and logistic regression. The proposed model is validated with 16 benchmark datasets and compared with EasyEnsemble, BalanceCascade, SMOTEBoost and RUSBoost methods. Ten-fold cross-validation is used for evaluating the performances in terms of overall accuracy and F-measure metric. Experimental results show that the proposed model is better than EasyEnsemble, BalanceCascade, SMOTEBoost and RUSBoost in both overall accuracy and F-measure metric.

This model can be used in any classification problem where class imbalance problem is a major concern. This model is very much useful in solving real-life problems. During pre-processing, TLRUS technique is used for undersampling; however, this technique can be made more useful

by improving detection of outlier and redundant pairs using some efficient data mining techniques.

Compliance with ethical standards

Conflict of interest None of the authors have the conflict of interest.

References

Bermejo P, Gamez JA, Puerta JM (2011) Improving the performance of Naive Bayes multinomial in e-mail foldering by introducing distribution-based balance of datasets. Expert Syst Appl 38(3):2072–2080

Biswas SK, Devi D, Chakraborty M (2017) A hybrid case based reasoning model for classification in Internet of Things (IoT) environment. J Organ End User Comput 30(4):104–122

Błaszczynski J, Deckert M, Stefanowski J, Wilk S (2010) Integrating selective pre-processing of imbalanced data with IVOTES ensemble. In: Rough sets and current trends in computing. Lecture Notes in Computer Science Series, vol 6086, Springer, pp 148–157

Cao D-S, Xu Q-S, Liang Y-Z, Zhang L-X, Li H-D (2010) The boosting: a new idea of building models. Chemom Intell Lab Syst 100(1):1–11

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357

Devi D, Biswas SK, Purkayastha B (2017) Redundancy-driven modified Tomek-link based undersampling: a solution to class imbalance. Pattern Recognit Lett 93:3–12

Elkan C (2001) The foundations of cost–sensitive learning. In: Proceedings of the 17th IEEE international joint conference on artificial intelligence (IJCAI'01), pp 973–978

Fradkin D, Muchnik I (2000) Support vector machines for classification. In: DIMACS series in discrete mathematics and theoretical computer science

Freund Y, Schapir RE (1996) Experiments with a new boosting algorithm. In: Machine learning: proceedings of the thirteenth international conference, pp 1–9

Galar M, Fernandez A, Barrenechea E, Bustince H, Herrera F (2012) A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans Syst Man Cybern Part C Appl Rev 42(4):463–484

Gunn SR (1998) Support vector machine for classification and regression. Techical report, University of Southampton, Southampton, UK

Khreich W, Granger E, Miri A, Sabourin R (2010) Iterative boolean combination of classifiers in the roc space: an application to anomaly detection with HMMS. Pattern Recognit 43(8):2732–2752

Kubat M, Holte RC, Matwin S (1998) Machine learning for the detection of oil spills in satellite radar images. Mac. Learn 30:195–215

Li C, Li J, He M (2016) Concept lattice compression in incomplete contexts based on K-medoids clustering. Int J Mach Learn Cybernet 7:539–552

Liu Y-H, Chen Y-T (2005) Total margin-based adaptive fuzzy support vector machines for multi-view face recognition. In: Proceedings of IEEE international conference on systems, man and cybernetics, vol 2, pp 1704–1711

Liu X-Y, Wu J, Zhou Z-H (2009) Exploratory undersampling for class imbalance learning. IEEE Trans Syst Man Cybern Part B Cybern 39(2):539–550

Mazurowski MA, Habas PA, Zurada JM, Lo JY, Baker JA, Tourassi GD (2008) Training neural network classifiers for medical decision making: the effects of imbalanced datasets on classification performance. Neural Netw 21(2–3):427–436

Mustafa G, Niu Z, Yousif A, Tarus J (2015) Solving the class imbalance problems using RUSMultiBoost Ensemble. In: Proceedings of 10th Iberian conference on information systems and technologies

Pallara A (1992) Binary decision tree approach to classification: a review of CART and other methods with some applications to real data. Statist Appl Ital J Appl Stat 4(3):1–32

Park H-A (2013) Introduction to logistic regression: from basic concepts to interpretation with particular attention to nursing domain. J Korean Acad Nurs 43(2):154–164

Phung S, Bouzerdoum A, Nguyen GH (2009) Learning pattern classification tasks with imbalanced data sets. In Yin (Eds.) Press. Pattern recognition, pp 193–208

Seiffert C, Khoshgoftaar TM, Hulse JV, Napolitano A (2010) RUSBoost: a hybrid approach to alleviating class imbalance. IEEE Trans Syst Man Cybern Part A Syst Hum 40(1):185–197

Sun Y, Wong AC, Kamel MS (2009) Classification of imbalanced data: a review. Int J Pattern Recognit 23(4):687–719

Sutton O (2012) Introduction to k-nearest neighbour classification and condensed nearest neighbour data reduction, pp 1–10

Tavallaee M, Stakhanova N, Ghorbani A (2010) Toward credible evaluation of anomaly-based intrusion-detection methods. IEEE Trans Syst Man Cybern Part C Appl Rev 40(5):516–524

Wang S, Yao X (2009) Diversity analysis on imbalanced data sets by using ensemble models. IEEE Symp Comput Intell Data Min CDIM'09:324–331

Yang Z, Tang W, Shintemirov A, Wu Q (2009) Association rule mining based dissolved gas analysis for fault diagnosis of power transformers. IEEE Trans Syst Man Cybern C Appl Rev 39(6):597–610

Zhu Z-B, Song Z-H (2010) Fault diagnosis based on imbalance modified kernel fisher discriminant analysis. Chem Eng Res Des 88(8):936–951

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.