# Improvement in Boosting Method by Using RUSTBoost Technique for Class Imbalanced Data


**Ashutosh Kumar, Roshan Bharti, Deepak Gupta and Anish Kumar Saha**

**Abstract** Class imbalance problem is common in many fields, and it occurs due to imbalanced dataset. A dataset is considered as imbalanced when number of examples in one class is more or less compared to another class. Data mining algorithms may generate suboptimal classification models when trained with imbalanced datasets. Several techniques have been proposed to solve the class imbalance problem. One of them includes boosting which is combined with resampling technique. It has gained popularity to solve class imbalance problem, for example, Random Undersampling Boosting (RUSBoost) and Synthetic Minority Oversampling Boosting Technique (SMOTEBoost). RUSBoost method uses random undersampling technique as resampling technique. One of the disadvantages of random undersampling may include loss of important data which is overcome by redundancy-driven modified Tomek-link based undersampling. A new hybrid undersampling algorithm is proposed in which we use redundancy-driven modified Tomek-link based undersampling as resampling technique along with boosting for learning from imbalanced training data. Experiments are performed for various datasets which are related to various application domains. The results are compared with decision tree, Support Vector Machine (SVM), logistic regression, and K-Nearest Neighbor (KNN) to check the performance of proposed method.

**Keywords** RUSBoost · SMOTEBoost · Class imbalance problem
Boosting technique · SVM · Decision tree

A. Kumar · R. Bharti · D. Gupta (✉) · A. K. Saha
Computer Science and Engineering, National Institute of Technology, Yupia, Arunachal Pradesh, India
e-mail: deepakjnu85@gmail.com

A. Kumar
e-mail: ashutoshsingh227@gmail.com

R. Bharti
e-mail: giet12cse077@gmail.com

A. K. Saha
e-mail: anishkumarsaha@gmail.com



J. Kalita et al. (eds.), *Recent Developments in Machine Learning and Data Analytics*,
Advances in Intelligent Systems and Computing 740,
https://doi.org/10.1007/978-981-13-1280-9_5

## 1 Introduction

Class imbalance [1] is a problem that occurs when the number of instances in one class is different compared to another class. The data preprocessing method includes two techniques, i.e., random oversampling and random undersampling. Random oversampling is applied in minority class, whereas random undersampling is applied in majority class until both the classes become equal. Major drawback of random oversampling is that the sampled training dataset is more as compared to the original training dataset, which is called as overfitting problem and also training time of the dataset is more as compared to original dataset in oversampling. Random undersampling method removes instances from majority class randomly until both the classes become equal. Since the number of instances in sampled training dataset is less compared to original training dataset, the training time of the classifier is also decreased. One of the major drawbacks of random undersampling is that some important instances may be deleted when we are removing instances from the majority class. SMOTEBoost [2] is a combination of Adaptive Boosting (AdaBoost) [3] and Synthetic Minority Oversampling TEchnique (SMOTE) [4]. SMOTE is an intelligent oversampling method which is specially designed for the treatment of imbalanced data problem. SMOTE is introduced to create new minority class instances by selecting some of the minority class instances which are nearest neighbor of each other. SMOTE method takes the difference between an instance of minority class and nearest neighbors, and then their difference is multiplied by a random number from 0 to 1 and the result is added to sample space under consideration. Since SMOTE method does not add the same instance from minority class. So, overfitting problem is obviated. Here, the data preprocessing (oversampling of minority class) is done by producing synthetic examples and the oversampled training dataset is given to AdaBoost to make a model. AdaBoost is most commonly used practical boosting algorithms that use entire dataset for training of the weak learners but during the training it gives more attention to incorrectly classified pattern. The importance of instances is calculated by a weight, which is initially same for all instances. Weights of misclassified patterns are increased, whereas the weight of actual classified pattern is decreased after each step or iteration. Boosting improves the accuracy of model by giving attention to examples, and at the same time SMOTE improves the performance of the model by increasing number of minority class instances. The only drawback associated with this method is that SOMTE is complex and time-consuming oversampling technique compared to other resampling methods. RUSBoost [5] is another technique which is a combination of random undersampling and AdaBoost. Random undersampling is applied to majority class examples to achieve desired balance ratio between minority and majority class, and then the balanced dataset is given to AdaBoost to build a model. The only drawback associated with RUSBoost is that the random undersampling technique removes the instances randomly from majority class which may discard instances not useful for classification. The performance is increased in RUSBoost compared to SMOTEBoost but still there is a loss of important data which is required by the classifier. A pair of instances is said to be

Tomek-link pair [6] if they are nearest neighbor of each other and belong to different classes. In this method, undersampling is performed by removing all Tomek-link pair from the training datasets.

Several techniques have been introduced to solve the class imbalanced problem. Galar et al. [7] provide some methods to manage class imbalance problem. They are classified into some categories based on how they deal with class imbalance problem. The first practical and most widely used boosting algorithm was AdaBoost algorithm. Hoang et al. [8] proposed a recognition-based approach, an alternative solution of class imbalance problem in which the classifier is trained with the examples of minority class in the absence of majority class examples. However, recognition-based approach is not applicable for classification algorithms such as decision tree, Naive Bayes, and associative classifier. Dennis L. Wilson proposed Edited Nearest Neighbor (ENN) [9], an undersampling method in which undersampling on majority class is performed by removing instances whose class label does not match with majority of its k-nearest neighbors, i.e., undersampling is performed on majority class by removing outliers. Another undersampling method known as Condensed Nearest Neighbor (CNN) was proposed by P. Hart et al. [10]. The main idea behind this method is to select a subset of original training dataset. Since CNN uses undersampling which requires many scans over training dataset; therefore, it is slow compared to other methods. SMOTEBoost and RUSBoost were also introduced to solve class imbalance problem. Readers can apply boosting techniques on [11–14] to improve the classification performance in terms of accuracy, G-mean, and CPU utilization.

Hence, in this paper, we are combining both RUSBoost and redundancy-driven modified Tomek-link based undersampling method to overcome class imbalance problem, and the algorithm is named as RUSTBoost (redundancy-driven modified Tomek-link based random undersampling boosting technique). Here, undersampling is done by redundancy-driven modified Tomek-link pair and boosting is done by AdaBoost algorithm with different learners, i.e., SVM, decision tree, KNN, and logistic regression.

Here, Sect. 2 represents the work related to RUSBoost algorithm and redundancy-driven modified Tomek-link undersampling method in detail; Sect. 3 describes the proposed RUSTBoost method in detail. In Sect. 4, we have compared the proposed RUSTBoost method with RUSBoost and SMOTEBoost in terms of accuracy and G-mean. Section 5 concludes the paper along with future work.

## 2 Background and Related Work

In this section, we have discussed RUSBoost algorithm and redundancy-driven modified Tomek-link based undersampling method.

## 2.1 RUSBoost Algorithm

This section describes the algorithm for RUSBoost proposed by Seiffert et al. [5].

Let $A$ be the set of instances $(P_1, Q_1), \ldots, (P_n, Q_n)$ with minority class $Q^t \in B, |B| = 2$, $T$ is the number of classifiers in the boosting, $r$ where $1 \leq r \leq R$ and $h_r$ is the iteration for weak learner hypothesis.

Output from the procedure is $H(P) = \arg\max(Q \in B) \sum_{r=1}^{R} h_r(P, Q) * \log\left(\frac{1}{\beta_r}\right)$

Procedure

WeakLearn ():
**Begin**

1. Initialize $D_1(i) = \frac{1}{n} \forall i$
2. Repeat for $r = 1, 2, \ldots, R$.
   (a) Temporary training dataset $A'_r$ is created by applying random undersampling with weight distribution $D'_r$ on original training dataset.
   (b) Call WeakLearn, by providing $A'_r$ and their weights $D'_r$.
   (c) Hypothesis is obtained as $h_r : X \times B \rightarrow [0, 1]$.
   (d) Pseudo-loss is calculated (for $A$ and $D_r$) as

$$\in_r = \sum_{(i,Q):Q_i \neq Q} D_r(i)(1 - h_r(P_i, Q_i) + h_r(P_i, Q))$$

   (e) Weight update parameter is calculated by using $\beta_r = \frac{\in_r}{1-\in_r}$.
   (f) Update $D_r$ using $D_{r+1}(i) = D_r(i)\beta_r^{\frac{1}{2}(1+h_r(P_i,Q_i)-h_r(P_i,Q:Q\neq Q_i))}$
   (g) Normalize $D_{r+1}$: Let $Z_r = \sum_i D_{r+1}(i)$ then $D_{r+1}(i) = \frac{D_{r+1}(i)}{Z_r}$

## 2.2 Redundancy-Driven Modified Tomek-Link Based Undersampling

To reduce significance loss of information from training data during undersampling, Devi et al. [15] have proposed a Tomek-link and redundancy-based undersampling method where eliminations are done for outliers and noisy (Tomek-linked) patterns with greater redundancy and lower contribution factor, while undersampling was performed. The redundancy-driven modified Tomek-link boosting method works in the following three steps:

Step-1: Elimination of outliers from majority class,
Step-2: Detection of redundant pairs, and
Step-3: Elimination of redundant patterns.

In the first step, outliers are eliminated from majority class where outliers are determined based on given Tomek-link pair and a threshold value (r). Let $A$ be a dataset with m number of patterns and n number of attributes in each pattern. Let each pattern is characterized as $(\vec{P}_a, Q_b)$ where $\vec{P}_a$ is the input vector for $a$th pattern, denoted as $\vec{P}_a = (P_{a1}, P_{a2}, \ldots, P_{an}), 1 \leq a \leq m$. Let $Q_b = (Q_1, Q_2, \ldots, Q_c)$ is the designated class label out of c classes. Let us assume the following representation: $A_{m \times n}$ as imbalanced dataset, with m patterns and n attributes, $A_{\min}$ as set of minority class patterns, i.e., $A_{\min} = \{P_i | i = 1, 2, \ldots, f\}$, where $f$ is the total count of minority class instances, $A_{\text{maj}}$ as set of majority class patterns, i.e., $A_{\text{maj}} = \{P_j | j = 1, 2, \ldots, g\}$, where $g$ is the number of majority class patterns, $TL$ is the set of Tomek-linked patterns, $\text{indices}_{P_j}$ is the total minority patterns to which majority pattern $P_j$ has associated as Tomek-linked pair, $TL'$ is revised subset of $TL$ after removal of $P_j$ patterns determined as outliers, and $A'_{\text{maj}}$ is the set of majority class patterns after removing outliers. Input to the procedure is $A_{\min}$, $A_{\text{maj}}$ and threshold (r) to determine majority patterns $P_j$.

Then, the procedure to eliminate outliers from majority class is proposed as given below.

**Begin**

**Step-1**: Entire dataset $A_{m \times n}$ is divided into two subsets say $A_{\min}$ and $A_{\text{maj}}$.
**Step-2**: For each $P_i \in A_{\min}$, determine its k-nearest neighbor (By using KNN algorithm) from $P_j$ such that $P_j \in A'_{\text{maj}}$ by using Euclidean distance.
**Step-3**: $TL = \{(P_i, P_j) | P_i \in A_{\min}, P_j \in A_{\text{maj}}\}$ is the nearest neighbor pair of $(P_i, P_j)$.
**Step-4**: For each $P_j \in A_{\text{maj}}$, determine $\text{indices}_{P_j}$.
**Step-5**: If $\text{indices}_{P_j} > r$, then $P_j$ is treated as outlier in the minority region and the output is stored in subset OUTER as $\text{OUTER} = \{P_j | \forall P_j, \text{indices}_{P_j} > r\}$.
**Step-6**: $TL$ is modified to $TL'$ as $TL' = TL = \{\text{OUTER}\}$.
**Step-7**: Subset $A_{\text{maj}}$ is updated to $A'_{\text{maj}}$ as $A'_{\text{maj}} = A_{\text{maj}} - \{\text{OUTER}\}$

Patterns creating the set OUTER are named as outliers and it must be removed. $TL'$ contains remaining Tomek-link paired patterns.

Further, in step-2, redundant pairs are detected from the remaining Tomek-link paired patterns. Majority class patterns are detected by together considering redundancy and noise factors which is occupying the proximity of the decision boundary and high similarity with other majority patterns. Let $A'_{\text{maj}}$ represent a new set of majority class patterns and $RD$ represent a set of most redundant majority pair patterns for each $P_n$ where $P_n \in A'_{\text{maj}}$ and $TL\_RD$ represents the intersection of $TL'$ and $RD$.

Input to the procedure is $A'_{\text{maj}}$, $TL'$ and $k$, where $k$ is the number of redundant patterns to create $RD$. Set $k = 1$.

Now, the procedure to detect redundant pairs from the remaining Tomek-link paired patterns is given as follows:

**Begin**

**Step 1:** Redundant pair is determined from $A'_{\text{maj}}$ by using the ED, CB, and CS $\forall P_n \in A'_{\text{maj}}$ and stored in the subset $RD$ as

$$RD = \left\{(P_n, P_t) | \forall P_n, P_t \in A'_{\text{maj}} \text{ and } \text{sim}(P_n, P_t) = \text{maximum}\right\}$$

**Step 2:** A revised subset $TL_{RD}$ is created by the intersection of $TL'$ and $RD$ as

$$TL\_RD = \left\{P_n \cap A'_{\text{maj}} | P_n \in TL', P_n \in RD\right\}$$

Redundant majority patterns which are stored in set $TL\_RD$ are considered for next step.

Here, after detecting the redundant patterns, it is eliminated at the last step, i.e., step-3. Now, to eliminate a redundant majority pattern from a majority redundant pair $(P_u, P_v)$, where $P_u, P_v \in TL\_RD$ and a contribution factor $\text{Contri}_P$ is employed, which is determined as follows: $\text{Contri}_P = \frac{1}{N} \times \left\{\sum_{i=1}^{m}\left(\sum_{k=1}^{n} \ln\ f(P_{ik}|C_n)\right)\right\}$

where $N$ indicates the total number of patterns and $\ln f(P_{ik}|C_n)$ denotes the log-likelihood function. Hence, majority patterns reclined to the set $TL\_RD$ are calculated in terms of contribution factors and redundancy. Eliminations are done for patterns with lowest contribution factor and greater redundancy. Redundant eliminated patterns are stored in set $TL\_RD'$ by the following step:

$$TL\_RD' = \max\left(\frac{\text{sim}(P_u, P_v)}{\text{Contri}_{P_u}}, \frac{\text{sim}(P_u, P_v)}{\text{Contri}_{P_v}}\right); P_u, P_v \in TL\_RD$$

If two majority patterns are redundant, then the value with lesser contribution factor will be removed. Now, the refined dataset $A_R$ is obtained by the following steps.

i. Update $A'_{\text{maj}}$ to $A''_{\text{maj}}$ as $A''_{\text{maj}} = A'_{\text{maj}} - TL\_RD'$.
ii. $A_R = A_{\min} \cup A''_{\text{maj}}$

Here, for preprocessing, the similarity measures considered are Euclidian Distance (ED), City Block distance (CB), and Cosine Similarity (CS). Now, the formula to calculate ED, CB, and CS is as follows:

$$ED = \sqrt{\left\{\left(\left(P_{i1} - P_{j1}\right)^2\right) + \left(\left(P_{i2} - P_{j2}\right)^2\right) + \cdots + \left(\left(P_{in} - P_{jn}\right)^2\right)\right\}}$$

$$CB = \left|P_{i1} - P_{j1}\right| + \left|P_{i2} - P_{j2}\right| + \cdots + \left|P_{in} - P_{jn}\right|$$

$$CS = \text{sim}\,(P_i, P_j) = \cos\,(P_i, P_j) = \frac{\vec{P}_i.\vec{P}_j}{|\vec{P}_i| \times |\vec{P}_j|}$$

## 3 Proposed Redundancy-Driven Modified Tomek-Link Based Undersampling (RUSTBoost) Method

In this section, to overcome the drawback of RUSBoost which used random undersampling as resampling technique and AdaBoost as boosting technique, we have proposed a new hybrid sampling or boosting model which uses redundancy-driven modified Tomek-link based undersampling as undersampling technique along with AdaBoost as boosting technique to learn for imbalanced training data and named as RUSTBoost. It is designed to increase the performance and effectiveness in terms of accuracy and Geometric mean (G-mean).

The working of RUSTBoost method is shown as block diagram in Fig. 1. Here, original dataset is divided into two classes for preprocessing: the first one is majority class patterns and other one is minority class patterns. For majority class patterns, preprocessing is performed for undersampling using redundancy-driven modified Tomek-link undersampling algorithm as discussed above in related work.

After the preprocessing using Tomek-link is over, new training dataset is formed by combining the obtained refined majority class patterns and minority class patterns. New training dataset and its weight distribution both are now given to the boosting algorithm to be trained. Here, boosting is done by SVM, KNN, logistic regression, and decision tree. SVM is mainly used for regression and data classification which

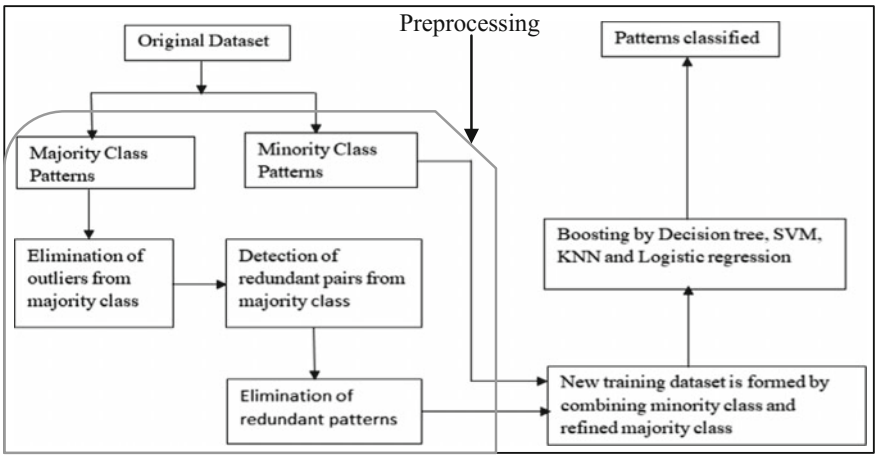


**Fig. 1** Block diagram of RUSTBoost

became commonly used classifier due to its theoretical concepts. The objective is to find a linear separating hyperplane and the equation is given as

$$wf(x) + b = 0$$

where $f(x)$ is a nonlinear function for higher dimension space, $w$ is the weight vector, and $b$ is the bias. In case of KNN, input is the K-nearest training samples and output is the class membership for classification and property value of object for regression, where regression is the process of estimating the relationship among variables. Logistic regression associates in analyzing the link between categorical-dependent variables and a variable quantity that predicts the likelihood of incidence of an occasion by fitting knowledge to a supply curve. Binary logistic regression and multinomial logistic regression are two models of logistic regression. Decision tree in boosting method is a binary tree like structure which is used for the logical representation of decisions and decision-making. The tree consists of root node, leaf node, and internal nodes. As per the condition given in the node, we will decide whether we will move to left or right.

Now, we will elaborate the RUSTBoost algorithm in detail.

RUSTBoost is mainly performed on the refined dataset $A_B$ but original dataset A is also required.

Input to the procedure is as follows: Set A of instances $(P_1, Q_1), \ldots, (P_n, Q_n)$ with minority class $Q^t \in B$, $|B| = 2$, T is the number of classifiers in the boosting, $r$ where $1 \leq r \leq R$, and $h_r$ is the iteration for weak learner hypothesis.

Output from the procedure is $H(P) = \arg\max(Q \in B) \sum_{r=1}^{R} h_r(P, Q) * \log\left(\frac{1}{\beta_r}\right)$

The procedure for RUSTBoost algorithm is given below:

WeakLearn():
**Begin**

1. Initialize $D_1(i) = \frac{1}{n} \forall i$
2. Repeat for $r = 1, 2, \ldots, R$
   (a) Create a temporary training dataset $A'_r$ by applying redundancy-driven modified Tomek-link based undersampling on original training dataset with weight distribution $D'_r$.
   (b) Call WeakLearn, by providing $A'_r$ and their weights $D'_r$.
   (c) Hypothesis is obtained as $h_r : X \times B \rightarrow [0, 1]$.
   (d) Pseudo-loss is calculated (for $A$ and $D_r$) as

$$\in_r = \sum_{(i,Q):Q_i \neq Q} D_r(i)(1 - h_r(P_i, Q_i) + h_r(P_i, Q))$$

   (e) Weight update parameter is calculated by using $\beta_r = \frac{\in_r}{1-\in_r}$.

(f) Update $D_r$ using $D_{r+1}(i) = D_r(i)\beta_r^{\frac{1}{2}(1+h_r(P_i,Q_i)-h_r(P_i,Q:Q\neq Q_i))}$
(g) Normalize $D_{r+1}$: Let $Z_r = \sum_i D_{r+1}(i)$, then $D_{r+1}(i) = \frac{D_{r+1}(i)}{Z_r}$

Here, in the first step, the weights of each pattern are initialized to $\frac{1}{n}$. In the second step, R weak hypotheses are iteratively trained. The undersampling algorithm is used to remove the majority class patterns in step 2a. As a result, $A'_r$ will have a new weight distribution $D'_r$. A weak learner creates the weak hypothesis $h_r$ in step 2c. Each hypothesis of a classifier is called as a weak learner. The final hypothesis $H(P)$ is returned as a weighted vote of the R weak hypotheses.

## 4 Experimental Results

We conducted experiments using 16 datasets downloaded from various application domains like the University of California, Irvine (UCI), and NASA [16, 17]. The detailed instruction about the downloaded datasets which is used to analyze the experimental result is given in Table 1.

We are using decision tree, SVM, KNN, and logistic regression as a base learner, and different evaluation metrics like accuracy and G-mean are considered. Our experiment shows that the proposed method (RUSTBoost) performs significantly better than RUSBoost and SMOTEBoost on benchmark datasets using F-measure and overall accuracy. After applying the algorithm, accuracy is being calculated for a total of 16 datasets and for all of the methods, i.e., SMOTEBoost, RUSBoost, and RUSTBoost, the difference in accuracy and G-mean is shown in Tables 2 and 3. Now, the graph for all the classifier, i.e., KNN, LR, DT, and SVM for accuracy and G-mean, is shown in Figs. 2, 3, 4, 5, 6, 7, 8, and 9.

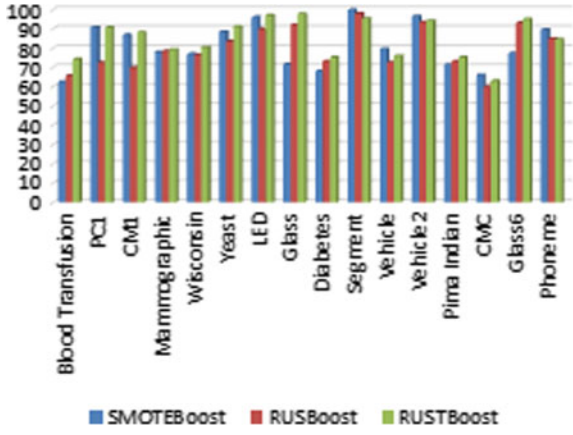


**Fig. 2** Accuracy of SMOTEBoost, RUSBoost, and RUSTBoost for decision tree

**Table 1** Detailed information about datasets

| S.No | Dataset | Minority class | Majority class | Instances | Attributes | No. of minority | No. of majority | Imbalance ratio (IR)* |
|---|---|---|---|---|---|---|---|---|
| 1 | Blood transfusion | 1 | 0 | 748 | 5 | 178 | 570 | 3.2022 |
| 2 | PC1 | 1 | 0 | 1109 | 22 | 77 | 1032 | 13.4025 |
| 3 | CM1 | 1 | 0 | 498 | 22 | 49 | 449 | 9.1632 |
| 4 | Mammographic | 1 | 0 | 1035 | 6 | 445 | 590 | 1.3258 |
| 5 | Wisconsin | 1 | 0 | 683 | 11 | 106 | 590 | 5.5660 |
| 6 | yeast | 1 | 0 | 1005 | 9 | 99 | 906 | 9.1515 |
| 7 | Led | 1 | 0 | 798 | 8 | 37 | 761 | 20.5675 |
| 8 | Glass | 1 | 0 | 897 | 10 | 60 | 837 | 13.95 |
| 9 | Diabetes | 1 | 0 | 768 | 9 | 268 | 506 | 1.8880 |
| 10 | Segment | 1 | 0 | 2308 | 20 | 329 | 1979 | 6.01519 |
| 11 | Vehicle | 1 | 0 | 846 | 19 | 217 | 846 | 3.8986 |
| 12 | Vehicle2 | 1 | 0 | 846 | 19 | 199 | 846 | 4.2512 |
| 13 | Pima Indian | 1 | 0 | 768 | 9 | 268 | 500 | 1.8656 |
| 14 | CMC | 1 | 0 | 850 | 10 | 208 | 642 | 3.0865 |
| 15 | Glass6 | 1 | 0 | 214 | 10 | 29 | 185 | 6.3793 |
| 16 | Phoneme | 1 | 0 | 5404 | 6 | 1586 | 3818 | 2.4073 |

**Table 2** Performance comparison of SMOTEBoost, RUSBoost, and RUSTBoost in terms of accuracy

| | Datasets | SMOTEBoost (KNN, LR, DT, SVM) (KNN, LR, DT, SVM) | RUSBoost (KNN, LR, DT, SVM) (KNN, LR, DT, SVM) | RUSTBoost (KNN, LR, DT, SVM) (KNN, LR, DT, SVM) |
|---|---|---|---|---|
| 1 | Blood transfusion | 57.27,64.79,62.34,63.55 | 64.74,71.41,65.55,77.14 | 69.15,76.87,74.21,76.88 |
| 2 | PC1 | 86.92,80.31,90.77,84.73 | 78.76,84.32,72.42,91.55 | 90.96,91.47,90.62,90.96 |
| 3 | CM1 | 82.13,86.19,86.77,81.77 | 75.35,79.74,69.71,87.57 | 84.36,88.18,88.18,89.97 |
| 4 | Mammographic | 73.78,78.58,77.79,81.77 | 77.33,79.54,78.41,77.84 | 77.16,78.05,79.06,77.55 |
| 5 | Wisconsin | 77.61,78.5,77.05,78.77 | 78.2,79.21,76.34,77.06 | 80.2,80.8,80.34,77.65 |
| 6 | Yeast | 85.88,78.23,88.36,86.44 | 87.06,87.72,83.48,89.82 | 91.15,91.13,91.04,90.55 |
| 7 | LED | 95.12,93.86,96.05,95.56 | 88.99,91.21,89.75,91.21 | 96.77,96.65,96.89,97.26 |
| 8 | Glass | 97.75,89.32,71.51,86.61 | 80.46,89.41,91.97,89.7 | 93.97,94.75,97.65,93.33 |
| 9 | Diabetes | 68.21,73.81,68.02,74.1 | 74.1,75.4,73.05,74.21 | 70.18,75.65,75.14,75.14 |
| 10 | Segment | 98.57,77.18,99.65,99.48 | 97.18,88.39,97.79,97.79 | 99,88.39,95.19,99.61 |
| 11 | Vehicle | 72.58,72.87,79.44,76.83 | 72.47,78.83,72.47,76.14 | 73.4,79.8,75.9,77.56 |
| 12 | Vehicle2 | 93.36,96.92,96.35,96.93 | 89.35,93.37,93.37,93.37 | 93.49,96.22,94.09,94.68 |
| 13 | Pima Indian | 69.41,75.53,71.36,75.53 | 73.17,75.66,73.18,75.8 | 70.18,75.65,75.14,75.14 |
| 14 | CMC | 62.63,57.59,66,57.59 | 54.35,61.52,60.01,64.11 | 59.22,65.67,62.9,67.48 |
| 15 | Glass6 | 94.49,96.35,77.27,96.19 | 90.51,92.5,93.01,93.41 | 92.22,94.2,95,92.5 |
| 16 | Phoneme | 88.71,72.87,89.6,73.83 | 86.05,74.87,84.84,76.92 | 85.14,73.93,84.48,76.28 |

**Table 3** Performance comparison of SMOTEBoost, RUSBoost, and RUSTBoost in terms of G-mean

| | Datasets | SMOTEBoost (KNN, LR, DT, SVM) (KNN, LR, DT, SVM) | RUSBoost (KNN, LR, DT, SVM) (KNN, LR, DT, SVM) | RUSTBoost (KNN, LR, DT, SVM) (KNN, LR, DT, SVM) |
|---|---|---|---|---|
| 1 | Blood transfusion | 0.23,0.24,0,0.33 | 0.11,0,0,0 | 0.24,0.32,0.24,0.34 |
| 2 | PC1 | 0.49,0.13,0.5,0 | 0.59,0,0.79,0.33 | 0.5,0.23,0.49,0.42 |
| 3 | CM1 | 0.42,0.01,0.44,0.44 | 0.43,0.72,0.58,0 | 0.44,0.45,0.44,0 |
| 4 | Mammographic | 0.56,0.62,0.54,0.64 | 0.6,0.6,0.9,0.1 | 0.67,0.65,0.63,0.67 |
| 5 | Wisconsin | 0.29,0.5,0,0.41 | 0.57,0.57,0.79,0.57 | 0.49,0.42,0.29,0.42 |
| 6 | Yeast | 0.72,0.41,0.64,0.6 | 0.72,0.49,0.93,0.82 | 0.59,0.51,0.71,0.76 |
| 7 | LED | 0.5,0.5,0.5,0.5 | 0.07,0,0.5,0 | 0.13,0.01,0,0.06 |
| 8 | Glass | 0.69,0.98,0.82,0 | 0.61,0.79,1,0 | 0.79,0.13,0.81,0 |
| 9 | Diabetes | 0.55,0.63,0.71,0.72 | 0.67,0.74,0.96,0.72 | 0.7,0.7,0.73,0.68 |
| 10 | Segment | 0.95,0.66,0.96,0.97 | 0.95,0.98,0.97,0.97 | 0.96,0.01,0.97,0.97 |
| 11 | Vehicle | 0.75,0.97,0.73,0.96 | 0.69,0.73,0.73,0.69 | 0.67,0.72,0.68,0.68 |
| 12 | Vehicle2 | 0.91,0.93,0.97,0.96 | 0.89,0.97,0.77,0.94 | 0.89,0.95,0.94,0.93 |
| 13 | Pima Indian | 0.68,0.71,0.68,0.75 | 0.65,0.72,0.69,0.94 | 0.7,0.7,0.73,0.68 |
| 14 | CMC | 0.56,0.58,0.34,0.64 | 0.49,0.53,0.57,0.54 | 0.49,0.57,0.57,0.62 |
| 15 | Glass6 | 0.72,0.95,0.79,0.69 | 0.65,0.69,0.79,0.69 | 0.72,0.64,0.82,0.69 |
| 16 | Phoneme | 0.54,0.71,0.84,0.71 | 0.84,0.7,0.8,0 | 0.83,0.72,0.8,0.72 |

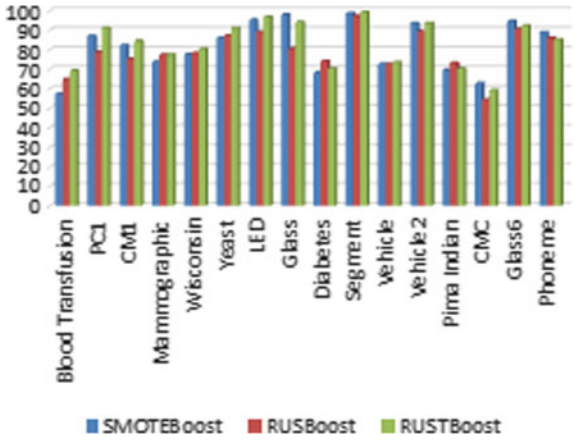


**Fig. 3** Accuracy of SMOTEBoost, RUSBoost, and RUSTBoost for KNN

The performance can be measured by parameters like accuracy and G-mean which can be calculated as

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \text{ and G - mean} = \sqrt{\frac{\text{TP}}{\text{TP} + \text{FN}} \times \frac{\text{TN}}{\text{TN} + \text{FP}}}$$

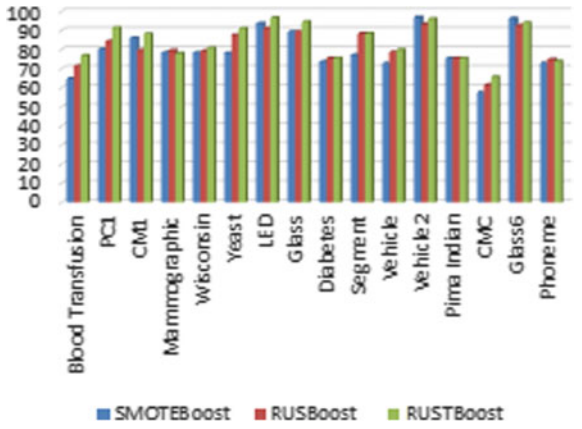


**Fig. 4** Accuracy of SMOTEBoost, RUSBoost, and RUSTBoost for logistic regression

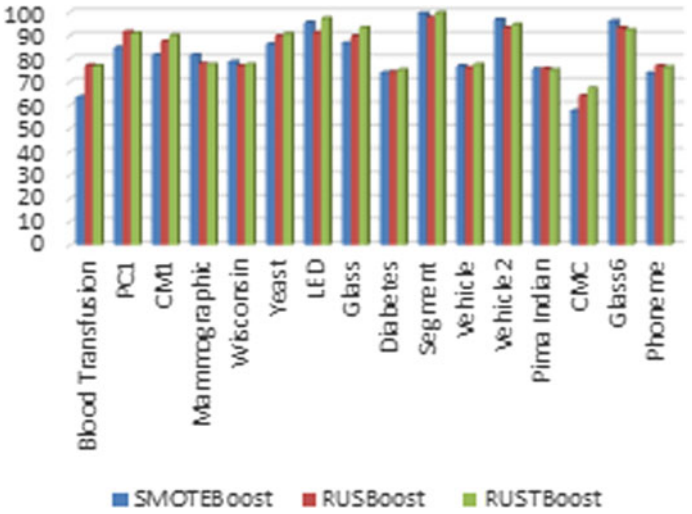


**Fig. 5** Accuracy of SMOTEBoost, RUSBoost, and RUSTBoost for SVM

Here, True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) can be defined by the following confusion matrix which contains info regarding actual and foreseen classifications done by a classifier.

| | | Predicted | |
|---|---|---|---|
| | | Positive | Negative |
| Actual | Positive | TP | FN |
| | Negative | FP | TN |

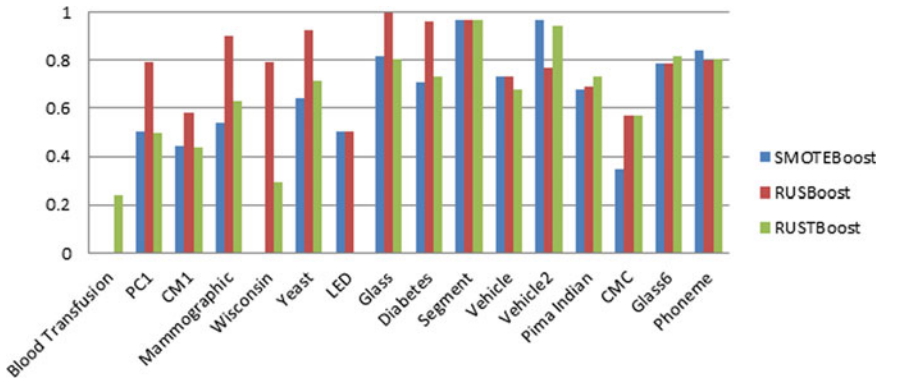


**Fig. 6** G-mean of SMOTEBoost, RUSBoost, and RUSTBoost for decision tree

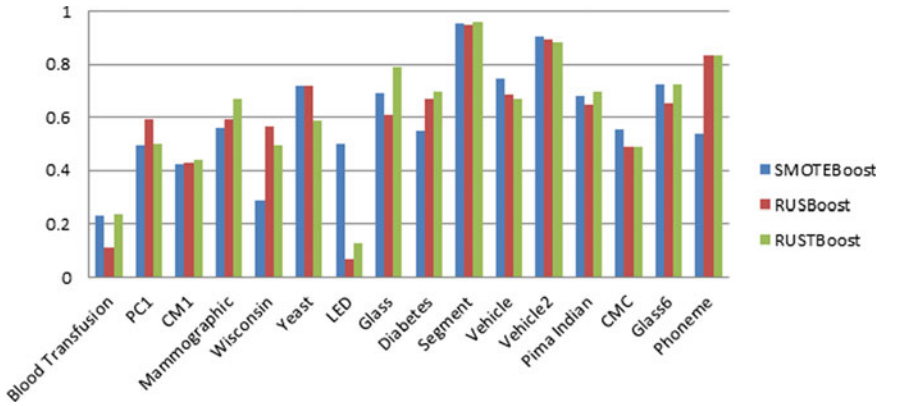


**Fig. 7** G-mean of SMOTEBoost, RUSBoost, and RUSTBoost for KNN

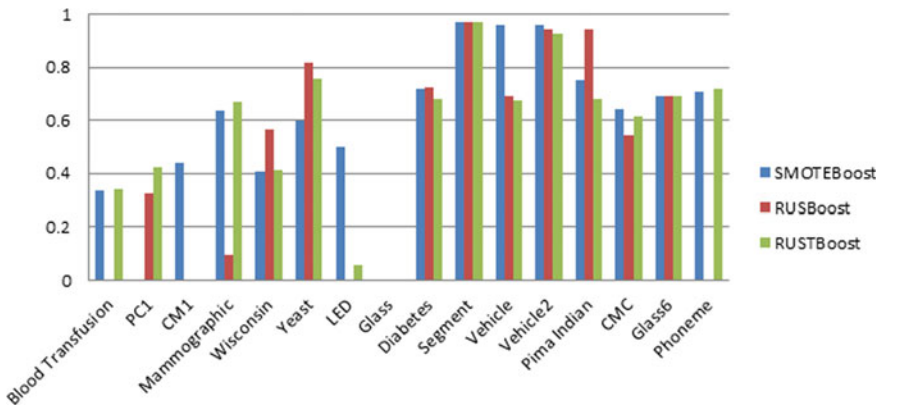


**Fig. 8** G-mean of SMOTEBoost, RUSBoost, and RUSTBoost for logistic regression

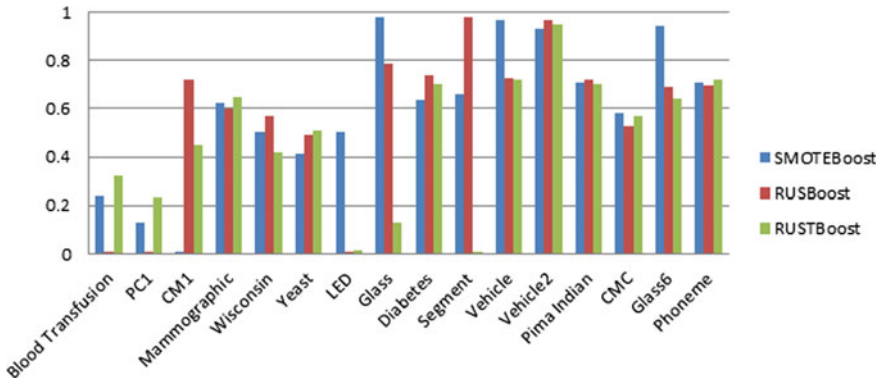


**Fig. 9** G-mean of SMOTEBoost, RUSBoost, and RUSTBoost for SVM

## 5 Conclusion and Future Work

In this paper, we have proposed RUSTBoost method which performs significantly better than RUSBoost and SMOTEBoost to overcome the loss of information associated with RUSBoost. It is simple and faster approach similar to RUSBoost for learning from training datasets where class distribution is unequal. It is also well suitable for noisy imbalanced datasets. In future, we may further improve the accuracy and G-mean of RUSTBoost by using other learner and boosting methods. Since the proposed method is only for binary class, one can also design the same for multiclass problem.

## References

1. Nyguyen, G.H., Bouzerdoum, A., Phung, S.L.: Learning pattern classification tasks with imbalanced data sets. University of Wollongong, Australia (2009)
2. Chawla, N.V., et al.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–327 (2002)
3. Yoav, F., Schapire, R.E.: Experiments with a new boosting algorithm, machine learning. In: Proceedings of the Thirteenth International Conference (1996)
4. Chawla, N.V., Lazarevic, L., Hall, O., Bowyer, K.W.: SMOTEBoost: improving prediction of the minority class in boosting. Proc. Knowl. Discov. Databases **2838**, 107–119 (2003)
5. Seiffert, C., Khoshgoftaar, T., Van Hulse, J., Napolitano, A.: Rusboost: A hybrid approach to alleviating class imbalance. IEEE Trans. Syst. Man Cybern. A Syst. Humans **40**(1), 185–197 (2010)
6. Tomek, Ivan: The modifications of CNN. IEEE Trans. Syst. Man Cybern. **6**, 769–772 (1976)
7. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., Herrera, F.: A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans. Syst. Man Cybern. C Appl. Rev. **42**(4), 463–484 (2011)
8. Nguyen, G.H., Bouzerdoum, A., Phung, S.: Learning pattern classification tasks with imbalanced data sets. In: Yin, P. (ed.) Pattern recognition, pp. 193–208. In-Tech, Crotia (2009)

9. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans. Syst. Man Cybern. **3**, 408–421 (1972)
10. Hart, P.: The condensed nearest neighbor rule. IEEE Trans. Inf. Theory **14**(3), 515 (2016)
11. Balasundaram, S., Gupta, D.: On optimization based extreme learning machine in primal for regression and classification by functional iterative method. Int. J. Mach. Learn. Cybernet. **7**(5), 707–728 (2016)
12. Tanveer, M., Khan, M.A., Ho, S.S.: Robust energy-based least squares twin support vector machines. Appl. Intell. **45**(1), 174–186 (2016). https://doi.org/10.1007/s10489-015-0751-1
13. Gupta, D., Borah, P., Prasad, M.: A fuzzy based Lagrangian twin parametric-margin support vector machine (FLTPMSVM). In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, pp. 1–7 (2017) https://doi.org/10.1109/ssci.2017.8280964
14. Balasundaram, S., Gupta, D., Prasad, S.C.: A new approach for training Lagrangian twin support vector machine via unconstrained convex minimization. Appl. Intell. **46**(1), 124–134 (2017). Springer
15. Devi, D., Biswasb, S.K., Purkayasthac, B.: Redundancy-driven modified Tomek-link based undersampling: a solution to class imbalance. Pattern Recogn. Lett. **93**, 3–12 (2016)
16. http://www.ics.uci.edu/~mlearn/~MLRepository.html
17. Metrics data program: http://mdp.ivv.nasa.gov