Publisher version: R. Soleymani, E. Granger and Giorgio Fumera, "Progressive boosting for class imbalance and its application to face re-identification," Expert Systems with Applications, Volume 101, 2018, Pages 271-291, ISSN 0957-4174,
DOI: 10.1016/j.eswa.2018.01.023

Progressive Boosting for Class Imbalance and Its Application to Face Re-Identification

Roghayeh Soleymani[a,*], Eric Granger[a], Giorgio Fumera[b]

[a] *Laboratoire d'imagerie, de vision et d'intelligence artificielle, École de technologie supérieure Université du Québec, Montreal, Canada*
[b] *Pattern Recognition and Applications Group, Dept. of Electrical and Electronic Engineering University of Cagliari, Cagliari, Italy*

Abstract

In practice, pattern recognition applications often suffer from imbalanced data distributions between classes, which may vary during operations w.r.t. the design data. For instance, in many video surveillance applications, e.g., face re-identification, the face individuals must be recognized over a distributed network of video cameras. An important challenge in such applications is class imbalance since the number of faces captured from an individual of interest is greatly outnumbered by those of others. Two-class classification systems designed using imbalanced data tend to recognize the majority (negative) class better, while the class of interest (positive class) often has the smaller number of samples. Several data-level techniques have been proposed to alleviate this issue, where classifier ensembles are designed with balanced data subsets by up-sampling positive samples or under-sampling negative samples. However, some informative samples may be neglected by random under-sampling and adding synthetic positive samples through up-sampling adds to training complexity. In this paper, a new ensemble learning algorithm called Progressive Boosting (PBoost) is proposed that progressively inserts uncorrelated groups of samples into a Boosting procedure to avoid loosing information while generating a diverse pool of classifiers. In many real-world recognition problems, the samples may be regrouped using some application-based contextual information. For example, in face re-identification applications, facial regions of a same person appearing in a camera field of view may be regrouped based on their trajectories found by face tracker. From one iteration to the next, the PBoost algorithm accumulates these uncorrelated groups of samples into a set that grows gradually in size and imbalance. Base classifiers are trained on samples selected from this set and validated on the whole set. Consequently, PBoost is more robust when the operational data may have unknown and variable levels of skew. In addition, the computation complexity of PBoost is lower than Boosting ensembles in literature that use under-sampling for learning from imbalanced data because not all of the base classifiers are validated on all negative samples. The new loss factor used in PBoost avoids biasing performance towards the negative class. Using this loss factor, the weight update of samples and classifier contribution in final predictions are set according to the ability of classifiers to recognize both classes. The proposed approach was validated and compared using synthetic data and videos from the Faces In Action, and COX dataset that emulate face re-identification applications. Results show that PBoost outperforms state of the art techniques in terms of both accuracy and complexity over different levels of imbalance and overlap between classes.

Keywords: Class Imbalance, Ensemble Learning, Boosting, Face Re-Identification, Video Surveillance.

1. Introduction

Class imbalance is a fundamental issue in many real-world pattern recognition applications found in, e.g., automated video surveillance, fraud detection, intrusion detection in computer and network security, risk management, and medical diagnosis. Imbalance appears in binary classification problems and binarization of multi-class classification problems using one-vs-all strategy when samples from one class are compared against all samples from all other classes (Galar et al., 2011; Wang & Yao, 2012). In particular, in face re-identification applications, systems for video-to-video face recognition are designed using faces of individuals captured from video sequences, and seek to recognize them when they appear in archived or live videos captured over a network of video cameras. Face tracking systems follow the position of faces over consecutive video frames and define the trajectories by collecting all face captures correspond to a same high quality track of an individual. Class imbalance is an important challenge in this application because the number of face captures from an individual of interest (positive class) may be greatly outnumbered by those of unknown or non-target individuals (negative class).

In practice, the level of imbalance observed during operations is unknown a priori and varies over time. This level of skew may differ from what is seen in the design data. Classification algorithms designed using imbalanced data are often bi-

*Corresponding author
Email addresses: rSoleymani@livia.etsmtl.ca (Roghayeh Soleymani), Eric.Granger@etsmtl.ca (Eric Granger), Fumera@diee.unica.it (Giorgio Fumera)

ased towards the majority (negative) class, even though the minority class is the (positive) class of interest. The main reason is that learning algorithms are typically designed to optimize the performance in terms of standard accuracy. Consequently, correct classification of negative class becomes their priority due to the abundance of samples for this class.

Several approaches have been proposed in literature to design ensembles of classifiers using imbalanced data (see the reviews (He & Garcia, 2009; Galar et al., 2012; Branco et al., 2016; Krawczyk, 2016; Haixiang et al., 2016)). In this paper, these approaches are divided into data-level and algorithm-level approaches. Data-level approaches either up-sample the positive class, under-sample the negative class or combine up-sampling and under-sampling to re-balance data for learning an ensemble of classifiers. Algorithm-level methods create or modify learning algorithms to counter the bias towards the negative class through cost-free techniques or by introducing uneven misclassification costs for the samples from different classes in cost-sensitive approaches.

Ensemble generation techniques can also be categorized into static and dynamic approaches. Static ensembles are designed a priori and face no change during operations. The ensembles selection or fusion may be set off-line using validation data, but typically assume a fixed level of imbalance during operations. Dynamic ensembles allow to adapt the selection and fusion of base classifiers during operations based on the operational data (Xiao et al., 2012; Galar et al., 2013a).

Most of the ensemble learning methods to handle imbalance in literature are static approaches. Boosting (Freund & Schapire, 1995; Freund et al., 1996) is a common static ensemble method that has been modified in several ways to learn from imbalanced data (see the reviews by (Galar et al., 2012; Branco et al., 2016; Krawczyk, 2016; Haixiang et al., 2016)). In data-level Boosting approaches, training data is rebalanced by up-sampling positive class, under-sampling negative class, or using both up-sampling and under-sampling (Chawla et al., 2003; Hu et al., 2009; Mease et al., 2007; Guo & Viktor, 2004; Seiffert et al., 2010; Galar et al., 2013b; Díez-Pastor et al., 2015). Up-sampling methods like SMOTEBoost (Chawla et al., 2003) are often more accurate, but they are computationally complex. In contrast, random under-sampling (RUS) (Seiffert et al., 2010) is more computationally efficient, but suffers from information loss. Partitional approaches (Soleymani et al., 2016a; Yan et al., 2003; Li et al., 2013) avoid information loss by splitting the negative class to uncorrelated subsets and training classifiers using all of these subsets.

Another issue with Boosting-based ensembles is that they may suffer from the bias of performance towards negative class because the loss factor, which guides their learning process, is obtained based on weighted accuracy. In cases of imbalance, weighted accuracy reflects the ability for correct classification of negative samples more than positive ones. This issue can be avoided by adopting a cost-sensitive approach (Fan et al., 1999; Ting, 2000; Sun et al., 2007), that defines different misclassification costs for different classes and integrates these cost factors into Boosting learning process. The drawback of these cost-sensitive techniques is that they rely on the suitable selection of cost factors which is often estimated by searching a range of possible values. In contrast, cost-free techniques modify learning algorithms by enhancing loss factor calculation without considering cost factors (Joshi et al., 2001; Kim et al., 2015; Soleymani et al., 2016b).

In literature, imbalance is addressed for face re-identification through dynamic and static approaches. The dynamic approaches base the selection and fusion of the classifiers on the estimated level of skew (Radtke et al., 2014; De-la Torre et al., 2015) to build robust ensembles of classifiers. In (Radtke et al., 2014; De-la Torre et al., 2015) authors design base classifiers for a range of different levels of imbalance for face re-identification in video surveillance application. Then, they estimate the skew level of input data stream and select a suitable fusion function based on that level. The level of imbalance may be difficult to estimate accurately during operations and the diverging selection and fusion function can decrease performance. In contrast, in a static approach (Soleymani et al., 2016a), the range of possible imbalance levels is accounted for during design by training base classifiers on data subsets with different imbalance levels.

In this paper we address the two above mentioned issues of Boosting-based ensembles in imbalanced problems, and in particular in face re-identification applications: the computational complexity of up-sampling methods, the loss of information of under-sampling ones; and the bias of the standard loss factor towards the negative class. To this aim we propose the Progressive Boosting (PBoost) algorithm to design static classifier ensembles that can maintain a high level of performance over a range of possible levels of imbalance and complexity in the data encountered during operations.

PBoost uses a partitioning method inspired by face re-identification applications, Trajectory Under-Sampling (TUS), that we proposed in our previous work [10]. TUS uses partitions of the negative class based on tracking information to design ensembles of classifiers. In particular, samples from the negative class are regrouped into disjoint partitions and, over iterations, these partitions are gradually accumulated into a temporary design subset. However, samples from the newly added partition and the important samples from previous iterations have an equally higher probability of being selected. The base classifier is then validated on the whole temporary subset. As with traditional Boosting ensembles, the samples that are misclassified are considered as the most important samples and their weights increase. With the sample selection scheme proposed in this paper, loss of information is considerably reduced, correlation among subsets of negative class is low, and only important samples tend to appear in more than one training subset. Therefore, the diversity and accuracy of Boosting ensembles tend to increase. In addition, to avoid biasing the performance towards the negative class, the proposed PBoosting algorithm employs a loss factor based on the F_β-measure, previously proposed by the authors (Soleymani et al., 2016b), that is applicable in any Boosting ensemble.

The diverse pool of classifiers generated with PBoost allows to globally model a range of different levels of imbalance and decision bound complexities for the data. Therefore, the static

ensembles produced using PBoost are robust to possible variations in data processed during operations because base classifiers are validated on a growing number of negative samples (imbalance level). In addition, the number of samples used per iteration to design (train and validate) a classifier in this ensemble is smaller than Boosting methods in the literature, which translates to a lower computational complexity for design.

The contributions of the proposed PBoost algorithm for face re-identification is summarized as follows:

- A new sample selection process for designing Boosting ensembles where negative class samples inter the Boosting process in uncorrolated partitions to avoid loss of information. Specifically for face re-identification application, the disjoint partitions are selected using tracking information.
- Modifying the validation step in Boosting learning such that base classifiers are validated on growing number of negative samples to increase robustness to imbalance and decrease computation complexity;
- Exploiting a specific loss factor (F-measure) in Boosting algorithm to avoid bias of performance towards the majority class.

The PBoost algorithm has been compared to state of the art Boosting ensembles on synthetic and video datasets, that emulate face re-identification application, in terms of both accuracy and computational complexity.

The rest of the paper is structured as follows. Section 2 contains a review of literature on ensemble learning for class imbalance in general and in face re-identification application. In Section 3, the proposed PBoost algorithm is described. The experimental methodology and results are presented in Sections 4 and 5, respectively.

2. Boosting Ensemble Learning for Class Imbalance

Learning from imbalanced data has been addressed in literature through data-level, algorithm-level, and cost-sensitive techniques. Ensemble learning methods exploit one or a combination of aforementioned techniques (Galar et al., 2012) to handle imbalance. Classifier ensembles can provide higher accuracy and robustness than a single classifier system by combining diverse classifiers (Rokach, 2010). Boosting is a common static ensemble learning algorithm initiated with AdaBoost (Freund & Schapire, 1995) and improved in AdaBoost.M1 (for 2-class problems) and AdaBoost.M2 (for multiple-class problems) (Freund et al., 1996) to effectively promote a weak learner that performs slightly better than random guessing into a stronger ensemble. In AdaBoost.M1 (Algo 1) samples are assigned weights that indicate their importance. These weights guide the learning process such that base classifiers in the ensemble focus on correct classification of more important samples as the learning iterations proceed. Samples that are misclassified in each iteration gain more importance for the next iteration and more accurate base classifiers gain higher contribution in final decision. These weights

Algorithm 1: AdaBoost.M1 ensemble learning method.

Input: Training set: $\mathbf{S} = \{(\mathbf{x}_i, y_i); i = 1,...,M\}, y_i \in \{-1,1\}$
# of iterations: E
Test input : $\mathbf{X}$
Output: Prediction Function: $H(\cdot)$

1 Initialize $\mathbf{W}_1(i) = \frac{1}{M}$ for $i = 1,...,M$.
2 **for** $e = 1,..,E$ **do**
- i Create new training set $\mathbf{S}'_e$ with weight distribution $\mathbf{W}'_e$.
- ii Train classifier C_e on $\mathbf{S}'_e$ with $\mathbf{W}'_e$.
- iii Test C_e on $\mathbf{S}$ and get back a label set $\{Y_i, i = 1,...,M\}$.
- iv Calculate the pseudo-loss for $\mathbf{S}$ and $\mathbf{W}_e$: $\varepsilon_e = \sum_{(i,Y_i):y_i \neq Y_i} W_e(i)$.
- v If $\varepsilon_e > 0.5$ **go to** step i
- vi Calculate the weight update parameter: $\alpha_e = \frac{\varepsilon_e}{1-\varepsilon_e}$
- vii Update $\mathbf{W}_{e+1}(i) = \mathbf{W}_e(i)\alpha_e^{|y_i - Y_i|/2}$
- viii Normalize $\mathbf{W}_{e+1}$ such that: $\sum \mathbf{W}_{e+1} = 1$.

3 Output the final hypothesis: $H(\cdot) = \sum_{e=1}^{E} h_e(\cdot) \log \frac{1}{\alpha_e}$

are used directly or for re-sampling training data, depending on the type of the base classifier being used. When the base classifier is from a type that is not designed to incorporate sample weights in its learning process (like SVMs), training data is re-sampled according to the weights of the samples. This case is considered here to explain the Boosting procedure.

Let's consider a two-class problem with M labelled training samples $\mathbf{S} = \{(\mathbf{x}_i, y_i); i = 1,...,M\}$ where $y_i \in \{-1,1\}$ that contains M^+ positive samples and M^- negative samples. All samples in the dataset are initially associated with the same weight $\mathbf{W_1}(i) = 1/M$, $i = 1,...,M$. Then, a new training subset is re-sampled into $\mathbf{S}'$ with $\mathbf{W}'$ to trained classifier C_e. This classifier is tested on all training samples ($\mathbf{S}$) and a loss factor (ε_e) is calculated as the sum of the weights of misclassified samples:

$$\varepsilon_e = \sum_{(i,Y_i):y_i \neq Y_i} \mathbf{W_e}(\mathbf{i}) \tag{1}$$

where Y_i is the label associated with $\mathbf{x}_i$ by C_e. If the classifier is too weak ($\varepsilon_e > 0.5$), the classifier is discarded and training set is re-sampled to train another classifier. The loss factor is then used to define a weight update factor:

$$\alpha_e = \frac{\varepsilon_e}{1-\varepsilon_e}. \tag{2}$$

The weights of the samples are then updated as:

$$\mathbf{W}_{e+1}(i) = \mathbf{W}_e(i)\alpha_e^{\frac{1}{2}|y_i - Y_i|}, \tag{3}$$

Weight vector is normalized such that the weights of the misclassified samples (more important samples) increase exponentially while the weights of the correctly classified samples decrease. α_e is also used to determine the contribution of the classifier in final predictions (Equation 4) so that more accurate

classifiers play more important role in identifying the class of the input sample. This process is repeated for a predefined number of times to design E classifiers. Considering $h_e(\mathbf{x})$ as the output of C_e (either a classification score or a label) for an input sample $\mathbf{x}$, final prediction of the ensemble is obtained from:

$$H(\mathbf{x}) = \sum_{e=1}^{E} h_e(\mathbf{x}) \log \frac{1}{\alpha_e} \quad (4)$$

Analogous to most learning algorithms, AdaBoost is not effective to learn from imbalanced data for two reasons. Negative samples are the majority and when training data is re-sampled in line 2.i of AdaBoost (see Algo. 1), they contribute more in $\mathbf{S}'$. Therefore, C_e is trained biased to correct classification of this class. After that, when C_e is tested on $\mathbf{S}$, loss factor in line 2.iv is calculated as a weighted error rate of classification. Again, negative samples contribute more in loss factor calculation and the weight update formula and classifiers contribution in final prediction become biased such that weight of negative samples increases for the next iteration and classifiers that mostly classify negative samples correctly get higher importance in final prediction of the ensemble. A taxonomy of methods in literature that modify AdaBoost to handle imbalance is presented in Figure 1. Based on the issue these approaches address, they are divided to two categories, data-level and algorithm-level methods that are presented in subsections 2.1 and 2.2, respectively.

2.1. Data-Level Methods:

Class imbalance can be handled in Boosting ensembles through up-sampling the positive class, under-sampling the negative class or combination of them. A popular up-sampling Boosting approach is SMOTEBoost (Chawla et al., 2003) that integrates Synthetic Minority Over-sampling Technique (SMOTE) into AdaBoost.M2. SMOTE creates synthetic samples by interpolating each positive sample with its k-nearest neighbours. MSMOTEBoost (Hu et al., 2009) use modified SMOTE (MSMOTE) by eliminating noisy samples and over-sampling only safe samples. Jous-Boost (Mease et al., 2007) oversample the positive class by duplicating it, instead of creating new samples, and introduce perturbation (jittering) to this data in order to avoid overfitting. DataBoost-IM (Guo & Viktor, 2004) oversample difficult samples from both classes and integrates it into AdaBoost.M1 .

Up-sampling techniques address the bias of performance in classifiers through balancing class distribution without loss of information. However, up-sampling, in general, increase the number of samples and consequently increase the complexity of learning algorithms, and SMOTE involves additional computations due to interpolating each sample with its k-nearest neighbours to generate synthetic samples.

In under-sampling Boosting category, RUSBoost (Seiffert et al., 2010) integrates random under-sampling (RUS) into AdaBoost.M1. RUSBoost is similar to AdaBoost presented in Algo. 1 where in line 2.i of this algorithm, $\mathbf{S}'$ contains all positive samples and a randomly selected subset of negative class, often with a size equal to the positive class. The subsets of negative class selected randomly over iterations of RUSBoost could be highly correlated and the classifiers trained on them can lack in diversity, especially when the skew level of training data is high. The sample selection paradigm in RUSBoost is managed in EUSBoost (Galar et al., 2013b) to create less correlated subsets using evolutionary prototype selection (García & Herrera, 2009).

Some researchers combine SMOTE and RUS in AdaBoost to achieve greater diversity and avoid loss of information as in Random Balance Boosting (RB-Boost) (Díez-Pastor et al., 2015). RB-Boost combines SMOTE and RUS to create training subsets with random and different skew levels in AdaBoost.M1.

Repetition of sampling in Boosting ensembles increase the chance of low correlation between subsets of data that are used for designing base classifiers and therefore maintain diversity among them. However, some potentially informative samples may be overlooked from these subsets in under-sampling process. In partitional approaches (Soleymani et al., 2016a; Yan et al., 2003; Li et al., 2013) bootstraps are selected without replacement either randomly (Yan et al., 2003), by clustering (Li et al., 2013) or based on a prior knowledge from the application (like trajectories in video surveillance applications such as face re-identification (Soleymani et al., 2016a)). In these ensemble bootstraps are drawn from a set of negative samples that reduces size in each iteration. In other words, after selection of a bootstrap in each iteration, its samples are eliminated from the main set. In random partitioning of negative samples by Yan et al. (Yan et al., 2003) the negative data is randomly decomposed into a number of subsets and each subset, combined with the positive samples, is used to train a classifier. Li et al. (Li et al., 2013) partition negative data by clustering it using k-means in the feature space and then create an ensemble from the classifiers trained on each negative cluster and the positive samples. The contribution of the classifiers in the ensemble are then weighted based on the distance between the corresponding negative cluster and positive class. In (Soleymani et al., 2016a), partitioning negative class is done by selecting samples from a set of trajectories that are formed based on the tracking information, as found in several video surveillance applications like face re-identification. In this approach, data from the trajectories are accumulated as the training iteration proceeds and therefore, base classifiers in the ensemble are trained on different imbalance levels to increase robustness of the ensemble to the possible variations in the skew level and complexity of operational data.

In contrast to RUSBoost, these partitional approaches use all negative samples from partitions to design ensembles and avoid loss of information. However, not all samples are informative and using all samples for training may result in unnecessary time and memory complexity. Therefore, enhancing partitional methods with more intelligent sample selection and ensemble learning algorithm (like RUSBoost) can avoid information loss and excessive time complexity at the same time.

2.2. Algorithm-Level Methods:

Using the standard loss factor based on misclassification rate in Boosting ensemble learning algorithms biases their performance towards negative class. In literature this issue is avoided

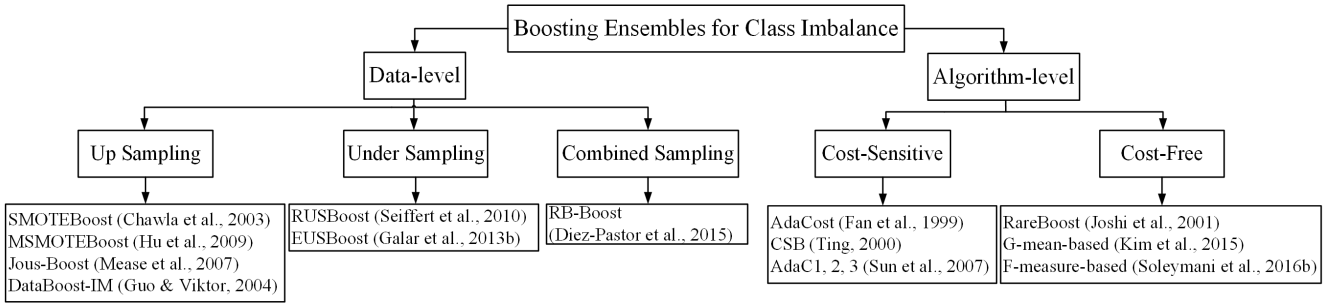


Figure 1: A taxonomy of Boosting ensembles learning methods specialized for imbalanced data.

at the algorithm level using two types of techniques; those that employ two different misclassification cost factors, one for positive and another for negative classes and those that handle this issue without the use of cost factors. Cost-sensitive Boosting methods including AdaCost (Fan et al., 1999), CSB (Ting, 2000) and AdaC (Sun et al., 2007), embed different misclassification cost factors into loss function or weight update formula of AdaBoost.M2.

Given μ_i as the cost factor of sample $\mathbf{x}_i$, in AdaCost (Fan et al., 1999), two cost adjustment functions are defined for each sample as $\phi_+ = -0.5\mu_i + 0.5$ and $\phi_- = 0.5\mu_i + 0.5$ and weight update formula is changed to:

$$\mathbf{W}_{e+1}(i) = \begin{cases} \mathbf{W}_e(i)\exp\{-\alpha_e\phi_+|y_i - Y_i|/2\} & \text{for } Y_i = 1 \\ \mathbf{W}_e(i)\exp\{-\alpha_e\phi_-|y_i - Y_i|/2\} & \text{for } Y_i = -1 \end{cases} \quad (5)$$

CSB (Ting, 2000) introduce two different cost factors for positive and negative classes as $\mu_+ = 1$ and $\mu_- \geq 1$, respectively.

$$\mathbf{W}_{e+1}(i) = \begin{cases} \mathbf{W}_e(i)\mu_+\exp\{-\alpha_e|y_i - Y_i|/2\} & \text{for } Y_i = 1 \\ \mathbf{W}_e(i)\mu_- exp(-\alpha_e|y_i - Y_i|/2\} & \text{for } Y_i = -1 \end{cases} \quad (6)$$

In AdaC1, 2, 3 (Sun et al., 2007) cost factors are embedded into the weight update formula in three different ways. Given $\mu_i \in [0, +\infty)$, in AdaC1:

$$\alpha_e = \frac{1}{2}\ln\frac{1 + \sum\limits_{i,y_i=Y_i}\mu_i\mathbf{W}_e(i) - \sum\limits_{i,y_i\neq Y_i}\mu_i\mathbf{W}_e(i)}{1 - \sum\limits_{i,y_i=Y_i}\mu_i\mathbf{W}_e(i) + \sum\limits_{i,y_i\neq Y_i}\mu_i\mathbf{W}_e(i)}, \quad (7)$$

$$\mathbf{W}_{e+1}(i) = \mathbf{W}_e(i)\exp\{-\alpha_e\mu_i Y_i y_i) \quad (8)$$

In AdaC2:

$$\alpha_e = \frac{1}{2}\ln\frac{\sum\limits_{i,y_i=Y_i}\mu_i\mathbf{W}_e(i)}{\sum\limits_{i,y_i\neq Y_i}\mu_i\mathbf{W}_e(i)}, \quad (9)$$

$$\mathbf{W}_{e+1}(i) = \mu_i\mathbf{W}_e(i)\exp\{-\alpha_e Y_i y_i\} \quad (10)$$

In AdaC3:

$$\alpha_e = \frac{1}{2}\ln\frac{\sum_i\mu_i\mathbf{W}_e(i) + \sum\limits_{i,y_i=Y_i}\mu_i^2\mathbf{W}_e(i) - \sum\limits_{i,y_i\neq Y_i}\mu_i^2\mathbf{W}_e(i)}{\sum_i\mu_i\mathbf{W}_e(i) - \sum\limits_{i,y_i=Y_i}\mu_i^2\mathbf{W}_e(i) + \sum\limits_{i,y_i\neq Y_i}\mu_i^2\mathbf{W}_e(i)}, \quad (11)$$

$$\mathbf{W}_{e+1}(i) = \mu_i\mathbf{W}_e(i)\exp\{-\alpha_e\mu_i Y_i y_i\} \quad (12)$$

In these cost-sensitive approaches by setting μ_+ greater than μ_- the weights of misclassified samples from positive class increase more than that of the misclassified samples from negative class. In addition, the weights of the classifiers that correctly classify positive class better than the negative class is higher in final decision. Therefore, these cost-sensitive approaches can make up for the usage of standard error rate in Boosting ensembles and allow adapting the performance by selecting proper cost factors based on the application. The drawback of these cost-sensitive approaches is that they require known μ_is that are usually set ad-hoc or by conducting a search in the space of possible costs for a dataset.

Some cost-free approaches have been proposed to deal with the bias of performance caused by using standard error in Boosting ensembles. In RareBoost (Joshi et al., 2001), two different αs are defined for positive and negative classes as:

$$\alpha_e^+ = \frac{1}{2}ln(\frac{TP_e}{FP_e}), \alpha_e^- = \frac{1}{2}ln(\frac{TN_e}{FN_e}) \quad (13)$$

where TP_e and TN_e are the true positive and true negative counts, respectively. Then the weight update formula and final classification prediction are modified as:

$$\mathbf{W}_{e+1}(i) = \begin{cases} \mathbf{W}_e(i)\exp\{-\alpha_e^+|y_i - Y_i|/2\} & \text{for } Y_i = 1 \\ \mathbf{W}_e(i)\exp\{-\alpha_e^-|y_i - Y_i|/2\} & \text{for } Y_i = -1 \end{cases} \quad (14)$$

$$H(\mathbf{x}) = sign(\sum_{e:h_e(\mathbf{x})\geq 0}\alpha_e^+ h_e(\mathbf{x}) + \sum_{e:h_e(\mathbf{x})<0}\alpha_e^- h_e(\mathbf{x}))) \quad (15)$$

Kim et al. (Kim et al., 2015) also define two different α_es for positive and negative classes as:

$$\alpha_e^+ = \frac{1 - l^+}{l^+}, l^+ = \frac{\sum\limits_{i;y_i=+1} W_e(i)|y_i - Y_i|/2}{\sum\limits_{i;y_i=+1} W_e(i)} \quad (16)$$

$$\alpha_e^- = \frac{1-l^-}{l^-}, l^- = \frac{\sum_{i;y_i=-1} W_e(i)|y_i - Y_i|/2}{\sum_{i;y_i=-1} W_e(i)} \quad (17)$$

where l^+ and l^- are pseudo errors of classifier in classifying each class. Finally:

$$\alpha_e = \ln(\sqrt{\mu_i \alpha_e^+ \alpha_e^-}) \ , \quad (18)$$

μ_i is a multiplier to control the weight of each sample.

Another cost-free approach (Soleymani et al., 2016b) modifies the loss factor calculation of Boosting algorithm using F-measure, the most frequently used measures for performance evaluation in class imbalance learning. To calculate this loss factor, the weight vector $\mathbf{W}_e$ is split to two weight matrices for positive $\mathbf{W}_e^+$ and negative $\mathbf{W}_e^-$ classes. Then, weighted versions of true positive, false positive, true negative and false negative counts are defined as:

$$\mathrm{TP}_e = \sum_{i:Y_i=1} \mathbf{W}_e^+(i), i = 1,...,M^+ \quad (19)$$

$$\mathrm{FP}_e = \sum_{i:Y_i=1} \mathbf{W}_e^-(i), i = 1,...,M^- \quad (20)$$

$$\mathrm{TN}_e = \sum_{i:Y_i=-1} \mathbf{W}_e^-(i), i = 1,...,M^- \quad (21)$$

$$\mathrm{FN}_e = \sum_{i:Y_i=-1} \mathbf{W}_e^+(i), i = 1,...,M^+ \quad (22)$$

Based on these values, the accuracy of a classifier is computed in terms of F_β-measure as:

$$A_F = \frac{(1+\beta^2)\mathrm{TP}_e}{(1+\beta^2)\mathrm{TP}_e + \mathrm{FP}_e + \beta^2 \mathrm{FN}_e} \ , \quad (23)$$

To measure the error of the classifiers, the corresponding loss factor is defined as:

$$L_e = 1 - A_F = \frac{\mathrm{FP}_e + \beta^2 \mathrm{FN}_e}{(1+\beta^2)\mathrm{TP}_e + \mathrm{FP}_e + \beta^2 \mathrm{FN}_e} \ . \quad (24)$$

The condition $\varepsilon_e > 0.5$ in line (v) of AdaBoost.M1 (Algo. 1) means that classifiers in a Boosting ensemble should perform better than random guessing. When F_β-measure is used as the evaluation metric, the base classifier to beat is the one that predicts everything as positive (Flach & Kull, 2015). Therefore, when the loss factor is calculated using Eq. (24), the accuracy criterion of 0.5 in AdaBoost.M1 should be replaced by $l_b = \frac{M^-}{(1+\beta^2)M^+ + M^-}$.

Cost-free methods enhance the performance of Boosting ensembles without setting any cost factors and guide the learning process using a more suitable loss factor calculation since the use of weighted standard accuracy, as in original Boosting algorithm, biases the learning process towards correct classification of the negative class. The problem with the loss factor proposed by Kim et al. (Kim et al., 2015) is that, if there are no misclassified samples in one class or in both classes, α_e is undefined. Besides, F_β-measure is more sensitive to imbalance than G-mean and at the same time allows us to give more importance to one class than the other. Therefore, this metric is used in loss factor calculation of the proposed PBoost algorithm.

2.3. Class Imbalance in Face Re-Identification

Face re-identification is a video surveillance application where video-to-video face recognition systems are designed to recognize faces of the individuals in archived or live videos at different time instants and/or locations over a network of distributed cameras. In this application, a face tracker defines facial trajectories for the moving faces captured over consecutive frames. An efficient tracking system does not mix the tracking information from several individuals between frames and therefore each trajectory corresponds to one individual. A trajectory is defined as a set of facial ROIs that correspond to a same high quality track of an individual across consecutive frames.

A common classification architecture in face re-identification is modular classification systems consisting of a single classifier or an ensemble of classifiers designed per target individual of interest. Two class classification systems in this application are designed using face captures from the target individual of interest (positive class) and those of the non-target individuals (negative class). This application is challenging due to variations in capture conditions such as pose, illumination, expression, etc. Moreover, an important challenge in this application is that the number of faces captured from the target individual is typically limited and greatly outnumbered by those of non-target ones. In addition, the level of imbalance during operations may differ from that of the design data.

There are some specialized approaches in literature to address imbalance in face re-identification application. In a dynamic approach (Radtke et al., 2014; De-la Torre et al., 2015), a pool of classifiers is generated using data subsets with different imbalance levels. These classifiers are then combined using Boolean combination and validated on different imbalance levels. During operations, the level of imbalance is estimated and the suitable Boolean function and its corresponding set of classifiers is selected.

The level of imbalance may be difficult to estimate accurately during operations and a static approach that accounts for variations in imbalance level of data may be of more interest. In the static approach of (Soleymani et al., 2016a), all the face captures are regrouped into trajectories. The classifiers are trained using a positive class trajectory and different numbers of negative class trajectories to design diverse and accurate classifiers. Selecting samples using trajectories to design classifier ensembles appears to be more effective than using the general-purpose sampling techniques (RUS and CUS) to improve accuracy. The same sample selection technique is utilized in the proposed PBoost algorithm for face re-identification application.

3. Progressive Boosting for Learning Ensembles from Imbalanced Data:

The Progressive Boosting (PBoost) learning method is proposed to sustain a high level of performance over a range of

imbalance and complexity levels in the data seen during operations. This method follows a static approach, and learns ensembles based on a combination of under-sampling and cost-free adjustment of Boosting ensemble learning.

With the PBoost algorithm, negative class is partitioned into disjoint subsets. These partitions are accumulated into a temporary design set progressively as learning iterations proceed. In each iteration, a subset of this temporary set is used for training a classifier such that the most important samples plus samples from the new partition are given an equally high opportunity to be used in training a base classifier. Loss of information is therefore avoided and ensemble diversity is increased. The trained classifier is then validated on the temporary set that contains all positive samples and only those negative partitions that have already been used in previous training iterations. As the temporary set grows, its imbalance level increases and therefore, the ensemble's robustness to diverse levels of skew and decision bound complexities during operations is increased. In PBoost, the error of the classifier is determined based on its ability to correctly classify both positive and negative classes. This loss factor plays an important role in determining the contribution of classifiers in final prediction, and in selection criteria of samples for designing the next classifiers.

There are several possible ways to partition the negative samples into disjoint subsets in literature (Xu & Wunsch, 2005) e.g., prototype-based methods like k-means and GMM algorithms, affinity-based methods like spectral, normalized-cut and subspace algorithms to represent the negatives, and thus define partitions (number of clusters and association of data to clusters). Two general-purpose partitioning techniques have been used in literature to partition data to learn ensembles from imbalanced data: Random Under-Sampling without replacement (we call RUSwR in this paper) (Yan et al., 2003) , and Cluster Under-Sampling (CUS) (Li et al., 2013). In some applications the data is already partitioned, like binarization of multi-class classification problems using one-vs-all strategy. In some others, the data may be grouped based on some contextual or application-based knowledge of data.

Trajectory Under-Sampling (TUS) is applicable in video surveillance applications where Regions Of Interest (ROIs), which are faces in face re-identification, are regrouped into a set called a trajectory with a high quality face tracker (Soleymani et al., 2016a). A high quality tracking system finds the trajectories by efficiently following the location of the ROIs that belong to the same individual over consecutive video frames. This application-based under-sampling method appeared to be more effective than the general-purposed under-sampling methods in designing classifier ensembles for face re-identification in terms of diversity and accuracy of opinions (Soleymani et al., 2016a).

The progressive Boosting method is presented in Algo. 2 and Figure 2. Its main steps are explained in the following. The negative samples are regrouped to E disjoint partitions $\mathbf{P}_e$, where $e = 1,..,E$, one per classifier in the ensemble (line 1). E and the number of negative samples in each partition N_e varies and depends on the partitioning method and the data distribution. In the case of random under-sampling without replacement E is preselected and N_e takes a fixed random value $N_e \in [M^+/2, 2M^+]$ such that $\sum_{e=1}^{E} N_e = M^-$. In the case of CUS and TUS, E and N_e depend on the number of samples that are assigned to each partition by the clustering algorithm and the tracker, respectively.

Given a training data set $\mathbf{S}$, one partition $\mathbf{P}_e$ is selected in each iteration and added to a temporary set $\mathbf{S}_e^{\text{tmp}}$ (line 5.ii) which initially contains the positive samples. The same initial weight w_{ini} is assigned to the samples in the new partition creating a weight vector $\mathbf{W}_e^p$ (line 5.i) which is also added to a temporary weight set $\mathbf{W}_e^{\text{tmp}}$ (line 5.ii). In the next step (line 5.iv), N_e samples from the temporary set $\mathbf{S}_e^{\text{tmp}}$ are selected through random under-sampling to create a new subset $\mathbf{S}_e^{'}$ with the weight distribution of $\mathbf{W}_e^{'}$. A classifier C_e is trained on $\mathbf{S}_e^{'}$ (line 5.v). Then it is tested on the whole temporary set $\mathbf{S}_e^{\text{tmp}}$ that has an imbalance level of $\lambda_e = 1 : \sum_{f=1}^{e} {}^{N_f}/_{M^+}$ (line 5.vi). Therefore, the classifiers in this ensemble are in fact validated on data subsets with a growing level of imbalance and complexity.

After that, the loss factor is calculated using the method proposed in (Soleymani et al., 2016b). The temporary weight vector $\mathbf{W}_e^{\text{tmp}}$ is split to two weight matrices for positive $\mathbf{W}_e^{\text{tmp},+}$ and negative $\mathbf{W}_e^{\text{tmp},-}$ classes. The size of $\mathbf{W}_e^{\text{tmp},+}$ is M^+ and the size of $\mathbf{W}_e^{\text{tmp},-}$ is $\sum_{f=1}^{e} N_f$, and:

$$\mathbf{W}_e^{\text{tmp},+} = \{\mathbf{W}_e^{\text{tmp}}(j), j = 1,...,(M^+ + \sum_{f=1}^{e} N_f) | y_j = 1\} , \quad (25)$$

$$\mathbf{W}_e^{\text{tmp},-} = \{\mathbf{W}_e^{\text{tmp}}(j), j = 1,...,(M^+ + \sum_{f=1}^{e} N_f) | y_j = -1\} . \quad (26)$$

Then, weighted versions of true positive, false positive, true negative and false negative counts are defined as:

$$\text{TP}_e = \sum_{k:Y_k=1} \mathbf{W}_e^{\text{tmp},+}(k), k = 1,...,M^+ \quad (27)$$

$$\text{FP}_e = \sum_{k:Y_k=1} \mathbf{W}_e^{\text{tmp},-}(k), k = 1,..., \sum_{f=1}^{e} N_f \quad (28)$$

$$\text{TN}_e = \sum_{k:Y_k=-1} \mathbf{W}_e^{\text{tmp},-}(k), k = 1,..., \sum_{f=1}^{e} N_f \quad (29)$$

$$\text{FN}_e = \sum_{k:Y_k=-1} \mathbf{W}_e^{\text{tmp},+}(k), k = 1,...,M^+ \quad (30)$$

To measure the error of the classifiers, the corresponding loss factor is defined as:

$$L_e = 1 - A_F = \frac{\text{FP}_e + \beta^2 \text{FN}_e}{(1+\beta^2)\text{TP}_e + \text{FP}_e + \beta^2 \text{FN}_e} . \quad (31)$$

After calculation of α_e (line 5.ix) from:

$$\alpha_e = \frac{L_e}{1 - L_e}, \quad (32)$$

the weights in the temporary set $\mathbf{W}_e^{\text{tmp}}$ are updated (line 5.x) as:

$$\mathbf{W}_{e+1}^{\text{tmp}}(j) = \mathbf{W}_{\text{e}}^{\text{tmp}}(j)\ \alpha_e^{|y_j - Y_j|/2} . \quad (33)$$

Even though it is desirable to limit the loss of information during under-sampling of data, some samples (like borderline samples) are of more interest than others for training classifiers in the ensemble. In Boosting ensembles, these samples are often detected as misclassified samples because borderline samples play more important role in defining the decision bound and they are more likely to be misclassified. More importance is given to these samples by assigning higher weights to them, so that they have a higher chance to be included in training subset(s). In the proposed PBoost ensemble, after normalization of $\mathbf{W}_e^{\text{tmp}}$, its maximum value among negative samples is selected as the initial weight for the next iteration (line 5.xii):

$$w_{e+1}^{\text{ini}} = \max_{y_j=-1} \{\mathbf{W}_e^{\text{tmp}}(j)\},\ j = 1,...,M^+ + \sum_{f=1}^{e} N_f. \tag{34}$$

This value corresponds to the weight of more important misclassified negative samples. Therefore, in each iteration, new samples and misclassified samples from previous iterations have more chance to be included in the training subset. Finally, α_e is used to obtain the final class prediction of the ensemble from (4) (line 6).

PBoost is somewhat inspired from RUSBoost, but differs in three main respects. First, during each iteration, instead of random under-sampling with replacement, most of training negative samples are selected from disjoint partitions. Consequently, repeatedly selection of the same samples over all iterations and information loss is avoided while the diversity increases. Second, instead of validating the classifiers on all samples, the classifiers are validated only on a subset of training set that grows in size and imbalance over iterations. Therefore, robustness to different levels of data imbalance and complexity increases, and the computations complexity of validation step decreases significantly. Third, instead of weighted accuracy, F-measure, an imbalance-compatible performance metric, avoids biasing performance towards negative class.

4. Experimental Methodology

In our experiments, the proposed PBoost ensemble learning method is assessed and compared with AdaBoost.M1 (Freund et al., 1996), and one state of the art method from each family of the data-level approaches reviewed in Section 2 including SMOTEBoost (Chawla et al., 2003), RUSBoost (Seiffert et al., 2010), and RB-Boost (Díez-Pastor et al., 2015). The datasets that are used for the experiments include: (1) A set of synthetic 2D data sets in which the level of skew and overlap between classes are controllable, (2) the Face In Action (FIA) video database (Goh et al., 2005) that emulates a passport checking scenario in face re-identification application, and (3) COX Face dataset the face recognition in video surveillance applications (Huang et al., 2015).

4.1. Datasets

4.1.1. Synthetic Dataset

The performance of classification systems may vary on different levels of overlap and skew between classes in both training and test data. Therefore, in our experiments on synthetic data, different synthetic datasets with different overlap and skew levels are generated and used to compare classification systems.

The data is generated to emulate both binarization of a multi-class classification problem when the classification strategy is one versus all and binary classification problems where there is no prior knowledge of optimal partitions. The samples of both positive and negative classes are generated from a mixture of Gaussian distributions. The samples from one normal distribution are considered as positive class and all other samples are considered as negative class.

To generate the 2D synthetic data, $M^+ = 100$ positive class samples are generated with a normal distribution as $N(m_+, \sigma_+)$, where $m_+ = (0,0)$ and $\sigma_+ = \left[\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right]$ indicate its mean and covariance matrix, respectively. Then, $T^- = 100$ points are generated randomly from a uniform distribution around m_+. These points $(m_{-,j},\ j = 1,...,T^-)$ are generated as the mean of T^- Normal distributions $(N(m_{-,j}, \sigma_-),\ j = 1,\ldots,T^-)$ for negative class where $\sigma_- = \sigma_+$. Each normal distribution contains $M^+ = 100$ samples and is considered as an ideal cluster of negative class (used for PCUS_i and to emulate face captures along trajectories in TUS). The mean of these clusters ($m_{-,j}$s) keep a margin distance δ from m_+. This margin is used to control the level of overlap between positive and negative classes.

For the experiments, we selected the parameter δ as 0.1 (maximum overlap) and 0.2 (medium overlap). For each overlap level, each normal distribution is randomly divided into two subsets for design and testing. Then the design subsets are divided into 5 folds considering one fold for validation and 4 folds for training. Five replications are carried out by alternating the validation fold in each iteration and by reversing the role of design and testing subsets for a total of 10 replications.

Two skew levels and two overlap levels of training data have been considered for the experiments, which have been combined into the three settings shown in Table 1. $\lambda_{\text{train}} = 1 : {}^{M^-}/_{M^+}$ is set to 1:50 in two setting and to 1:20 in the other. When $\lambda_{\text{train}} = 1:50$, only 50 clusters from the negative class are used for training. The objective is to compare different classification algorithms when they are designed on different levels of imbalance. Properties of training data generated with these settings are summarized in Table 1 and examples are presented in Figure 3.

In a similar way, four imbalance levels ($\lambda_{\text{test}} = \{1:1, 1:20, 1:50, 1:100\}$) are considered for testing to evaluate the robustness of the classification algorithms over varying skew levels of data during operation. Examples of synthetic test data corresponding to setting D_1 are presented in Figure 4.

4.1.2. Face Re-Identification Dataset

To compare the performance of proposed PBoost algorithm to state of the art classification systems, two datasets for video-based face recognition were considered.

FIA video database (Goh et al., 2005) contains video sequences that emulate a passport checking scenario. The video streams are collected from 221 participants under different capture conditions such as pose, illumination and expression in

Algorithm 2: Progressive Boosting ensemble learning method.

Input: Training set: $\mathbf{S} = \{(\mathbf{x}_i, y_i); i = 1, ..., M\}, y_i \in \{-1, 1\}, M = M^- + M^+$

Output: Predicted score or label: $H(\cdot)$

1 Partition non-target samples from $\mathbf{S}$ into E clusters $\{\mathbf{P}_e; e = 1, ..., E\}$.

2 Create a temporary training set and weight vector: $\mathbf{S}_1^{\text{tmp}} \leftarrow \{(\mathbf{x}_i, y_i) \in \mathbf{S} | y_i = 1\}$ and $\mathbf{W}_1^{\text{tmp}}(k) = 1, k = 1, ..., M^+$.

3 Initialize $w_1^{\text{ini}} = 1$.

4 Set $l_b = \frac{M^-}{(1+\beta^2)M^+ + M^-}$

5 **for** $e = 1, .., E$ **do**

i Initialize weight distribution of $\mathbf{P}_e$ as $\mathbf{W}_e^p(k) = w_e^{\text{ini}}, k = 1, ..., N_e$. // N_e is the size of $\mathbf{P}_e$.

ii $\mathbf{S}_e^{\text{tmp}} \leftarrow \mathbf{S}_e^{\text{tmp}} \bigcup \mathbf{P}_e$, $\mathbf{W}_e^{\text{tmp}} \leftarrow \mathbf{W}_e^{\text{tmp}} \bigcup \mathbf{W}_e^p$

iii Normalize $\mathbf{W}_e^{\text{tmp}}$ such that: $\sum \mathbf{W}_e^{\text{tmp}} = 1$.

iv Randomly select N_e non-target samples from $\mathbf{S}_e^{\text{tmp}}$ based on $\mathbf{W}_e^{\text{tmp}}$, to create a training subset $\mathbf{S}_e^{'}$ with $\mathbf{W}_e^{'}$.

v Train C_e on $\mathbf{S}_e^{'}$ with $\mathbf{W}_e^{'}$.

vi Test C_e on $\mathbf{S}_e^{\text{tmp}}$ and get back labels $Y_j, j = 1, ..., (M^+ + \sum_{f=1}^{e} N_f)$.

vii Calculate the pseudo-loss for $\mathbf{S}_e^{\text{tmp}}$ from $\mathbf{W}_e^{\text{tmp}}$ (using Equations 25 to 30): $\mathbf{W}_e^{\text{tmp},+} = \{\mathbf{W}_e^{\text{tmp}}(j), j = 1, ..., (M^+ + \sum_{f=1}^{e} N_f) | y_j = 1\}$,
$\mathbf{W}_e^{\text{tmp},-} = \{\mathbf{W}_e^{\text{tmp}}(j), j = 1, ..., (M^+ + \sum_{f=1}^{e} N_f) | y_j = -1\}$,
$\text{TP}_e = \sum_{(k,Y_k):Y_k=1} \mathbf{W}_e^{\text{tmp},+}(k), k = 1, ..., M^+$,
$\text{FP}_e = \sum_{(k,Y_k):Y_k=1} \mathbf{W}_e^{\text{tmp},-}(k), k = 1, ..., \sum_{f=1}^{e} N_f$,
$\text{TN}_e = \sum_{(k,Y_k):Y_k=-1} \mathbf{W}_e^{\text{tmp},-}(k), k = 1, ..., \sum_{f=1}^{e} N_f$,
$\text{FN}_e = \sum_{(k,Y_k):Y_k=-1} \mathbf{W}_e^{\text{tmp},+}(k), k = 1, ..., M^+$,
$L_e = 1 - A_F = \frac{\text{FP}_e + \beta^2 \text{FN}_e}{(1+\beta^2)\text{TP}_e + \text{FP}_e + \beta^2 \text{FN}_e}$.

viii If $L_e > l_b$ **go to** step iv

ix Calculate the weight update parameter: $\alpha_e = \frac{L_e}{1 - L_e}$

x Update $\mathbf{W}_{e+1}^{\text{tmp}}(j) = \mathbf{W}_e^{\text{tmp}}(j) \alpha_e^{|y_j - Y_j|/2}$

xi Normalize $\mathbf{W}_{e+1}^{\text{tmp}}$ such that: $\sum \mathbf{W}_{e+1}^{\text{tmp}} = 1$.

xii Set $w_{e+1}^{\text{ini}} = \max(\mathbf{W}_e^{\text{tmp}}), y_j = -1$

6 Output the final hypothesis: $H(\cdot) = \sum_{e=1}^{E} h_e(\cdot) \log \frac{1}{\alpha_e}$ // $h_e(\cdot)$ is the output of C_e.

Table 1: Settings used for data generation.

	D_1	D_2	D_3
λ_{tr}	1:50	1:50	1:20
δ	0.2	0.1	0.2

both indoor and outdoor environments. Videos were collected over three sessions where second and third sessions are three months later than the previous one. The participants are present before 6 cameras for about 5 seconds, resulting in total of 18 video sequences per person.

For experiments in this paper using FIA dataset, only the faces captured with frontal camera in indoor environment is used for both design and testing. ROIs are converted to gray-scale and rescaled to 70×70 pixels using Viola Jones algorithm (Viola & Jones, 2001) from this video. Some examples of ROIs from this data set are presented in Figure 5.

The COX Face dataset for face recognition in video surveillance (Huang et al., 2015) contains videos from 1000 participants captured with 4 cameras under different capture conditions. The faces are tracked and resized such that for each frame with a face detected, an image patch centered at the head of the subject is cropped out with a size of 66×66.

For experiments with video data, multi-resolution gray-Scale and rotation invariant Local Binary Patterns (LBP) (Ojala et al., 2002) histograms have been extracted as features. The local image texture for LBP has been characterized with 8 neighbours on a 1 radius circle centred on each pixel. Finally, a feature vector with the length of 59 has been obtained for each ROI.

10 individuals are randomly selected as targets and 91 individuals are randomly selected as non-targets. In each round of experiment, face patterns of one target individual (a trajectory) is considered as the positive class and 100 individuals (including 9 other target individuals and 91 non-target individuals) are

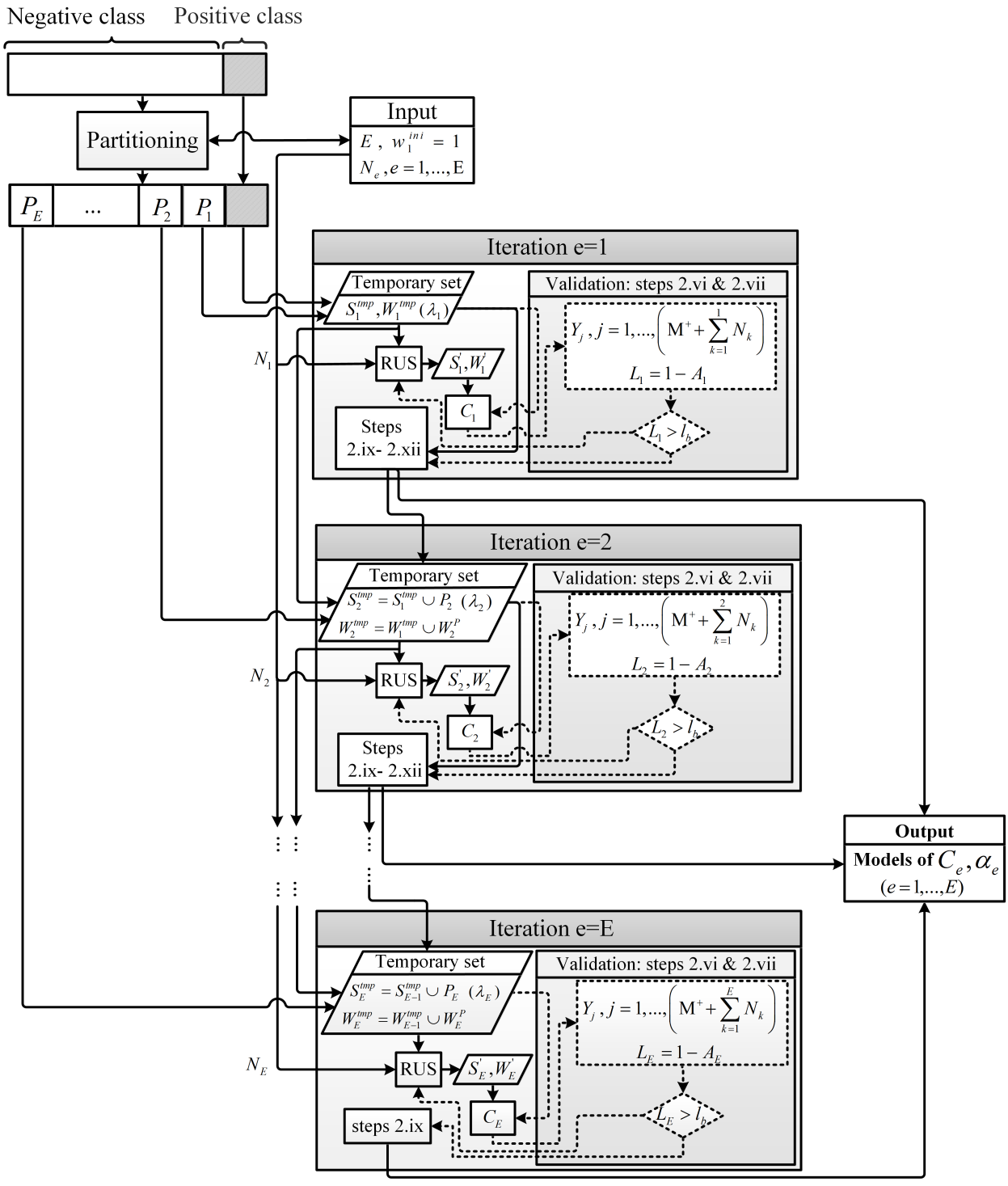


Figure 2: Block diagram representation of PBoost learning method.

selected as the negative class[1]. ROI patterns for each trajectory are divided into 2 sets for design and testing. The design set is divided to 5 folds, and for each round one fold is considered for validation and remaining 4 folds are considered for training. Then the roles of design and testing sets is reversed. Therefore, for each target individual, three independent sets are collected from these face patterns for training, validation and testing. Each set contains one group of samples from the target individual, 9 groups of samples from the remaining target

[1]The ROIs of each individual in FIA and COX datasets are already grouped.

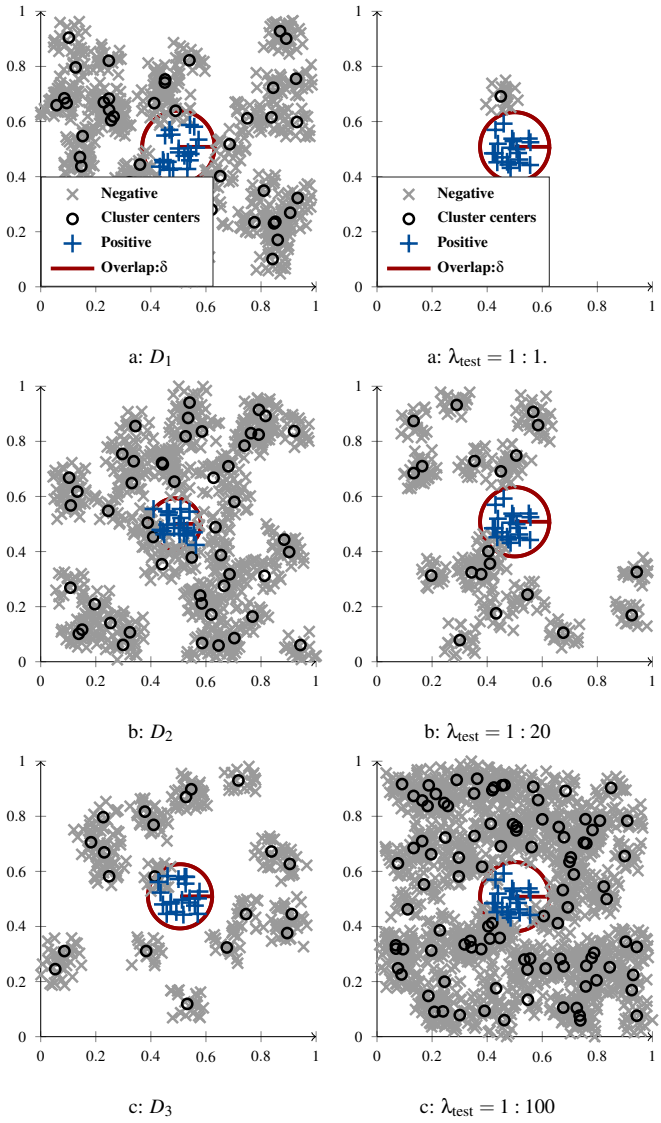


Figure 3: Examples of synthetic training data generated under different settings D_1, D_2 and D_3 .

Figure 4: Examples of synthetic test data generated with $\delta = 0.2$ and different skew levels λ_{test}.

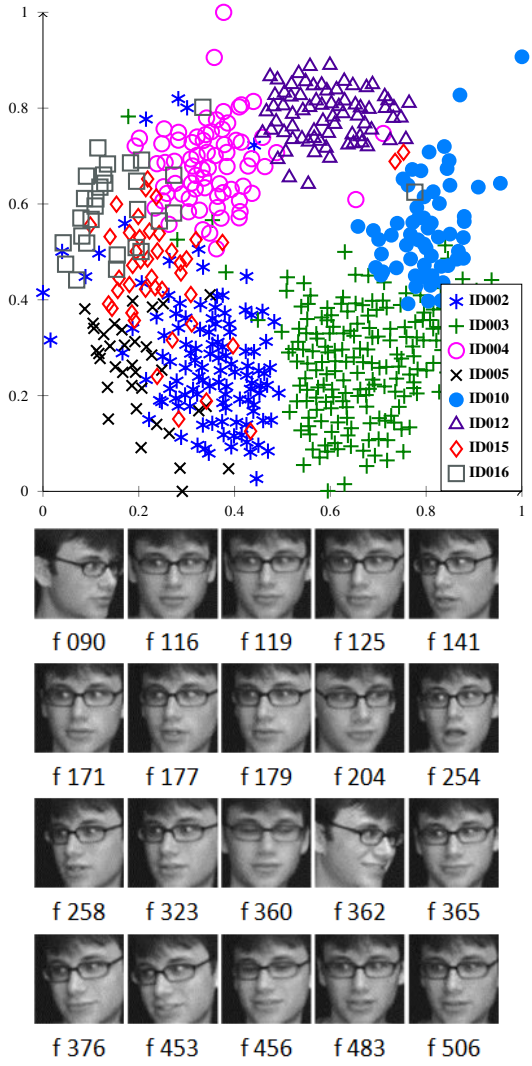


Figure 5: Examples of 2D mapping of LBP feature vectors belonging to 8 individuals using Sammon mapping (Sammon, 1969) on the left, and examples of 70×70 pixels ROIs along a trajectory captures with camera 3, during session one for ID010 with their frame numbers on the right.

individuals and 90 groups of samples from non-target individuals. Repeating this process for each target individual yields $10 \times 10 = 100$ overall experiments for this dataset.

Two imbalance levels ($\lambda_{train} = 1:50$ and 100) and four different imbalance levels $\lambda_{test} = \{1:1, 1:20, 1:50, 1:100\}$ are considered for selecting the training and testing negative class for each positive individual, respectively. This is to evaluate the performance of different classification algorithms when they are trained on different imbalance levels, and to evaluate the robustness of the classification algorithms over varying skew levels during operations. When $\lambda_{train} = 1:50$, for each positive individual, only $T^- = 50$ of 100 other individuals are used as the negative class from the training set that was collected for that positive individual. Therefore, when $\lambda_{test} = 1:100$, there are 50 negative individuals in the testing set that were not included in training the classification systems and the skew level of test data is higher than the skew level of training data. When $\lambda_{test} < 1:50$, most of the negative individuals that were used for training do not appear in testing data. When $\lambda_{train} = \lambda_{test} = 100$, the maximum imbalance level of testing data is the same as the imbalance level of training data. Therefore, all individuals are seen in both training and testing. However, in this case a high level of imbalance exists in both training and testing stages that makes both learning and classification more difficult. It is worth mentioning that in all settings, the skew level of the validation data is selected to be the same as testing data.

4.2. *Experimental Protocol*

For validation, synthetic and video datasets are used to evaluate the algorithm when the ideal partitions (or clusters) of nega-

tive class are known a priori. These data sets are also used for a binary classification problem where no information is available regarding the ideal clusters of data.

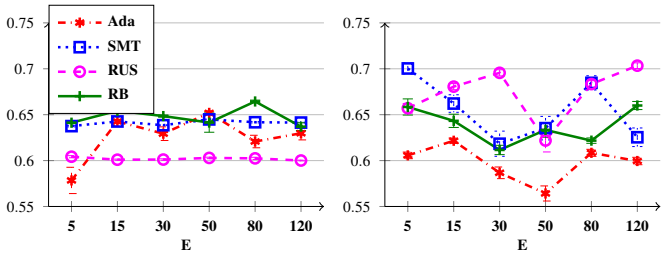
We use SVM with RBF kernel (Chang & Lin, 2011) as the base classifier where $K(\mathbf{x}', \mathbf{x}'') = \exp\{-\|\mathbf{x}'-\mathbf{x}''\|^2/2\kappa^2\}$. The kernel parameter κ is set as the average of the mean minimum distance between any two training samples and the scatter radius of the training samples in the input space (Li et al., 2008). The scatter radius is calculated by selecting the maximum distance between the training samples and a point corresponding to the mean of training samples. We used the LibSVM implementation of (Chang & Lin, 2011).

A brief description of the implemented ensembles, their variants and the abbreviations used for them are shown in Table 2. The last column of the table shows the datasets that are used for experiments on these classification systems. The abbreviations assigned to these ensembles are selected based on their sampling techniques and loss factor.

The baseline sampling techniques include Ada (resampling in AdaBoost), SMT (SMOTE in SMOTEBoost), RUS (random under-sampling in RUSBoost), RB (random balance in RB-Boost). For PBoost four partitioning techniques are used for under-sampling the negative class to evaluate the effect of the partitioning technique on the performance of PBoost ensemble: random under sampling without replacement (PRUS) and cluster under-sampling (PCUS) are used as general partitioning techniques for PBoost disregarding the data structure, whether or not the negative class is partitioned a priori. For PCUS, kernel k-means is used for clustering negative samples. To select k, it is varied over a range of possible values and the value of Dunn index (Dunn, 1973) is calculated for each case using a validation set. Finally, the optimal k, is selected when Dunn index takes its maximum value. Two cases are considered for PBoost in which the partitions of the negative class are known a priori. The ideal cluster under-sampling (PCUS_i) with synthetic datasets and trajectory under-sampling (PTUS) with video dataset.

The loss factor is calculated in two ways based on: the traditional technique i.e. weighted accuracy, and the F-measure. To indicate the use of F-measure in the Boosting ensembles in Table 2, the abbreviation is followed by -F. For the use of proposed loss factor calculation with the F-measure, β is set as 2 in all experiments because $\beta \geq 1$ is more suitable for imbalanced data classification when the positive class is the minority class. An experiment is done to evaluate the performance of Boosting ensembles with different values of β.

In addition to the mentioned Boosting ensembles, the ensembles proposed in (Soleymani et al., 2016a) for face re-identification are also included in the comparison. Rows 7 and 8 in Table 2 summarize the properties of these ensembles. In this technique, the negative class samples are regrouped to subsets with growing imbalance levels using CUS_i for synthetic data and TUS for video data. In contrast to Boosting ensembles, this method does not involve the use of any loss factors in learning process. However, the contribution of base classifiers in final prediction depends on their performance in terms of F-measure.



(a) AUPR. (b) F_2-measure.

Figure 6: Performance of baseline Boosting ensembles for different values of E on D_2 with $\lambda_{\text{test}} = 1:100$.

In the experiments with synthetic and video data sets, two different imbalance levels are used for training and four different imbalance levels are used for testing. This is to evaluate the sensitivity of classification systems to the level of imbalance during training and their robustness to possible variations in skew level during operations. In experiments with synthetic data, the overlap level between positive and negative classes are also varied because the issue of imbalance is related to the level of overlap between classes (López et al., 2013).

In experiments with synthetic and video datasets, the size of all Boosting ensembles is set equal to the maximum imbalance level of the data, except from PCUS. The reason for this setting is that the number of ideal clusters and the number of trajectories are both known and equal to the level of skew. In addition, based on a preliminary experiment under setting D_2 (see Table 1) on baseline ensembles in Figure 6, it is observed that the size of these ensembles does not have a significant impact on their performance. The performance of these ensembles vary in terms of F_2-measure as the ensemble size grows. However, their global performance in terms of AUPR do not change significantly. For PCUS, the size of ensemble is selected equal to the optimal k obtained using Dunn index.

4.3. Performance Evaluation

Global performance evaluation curves such as ROC and precision-recall, show the trade off between two metrics for different operational settings. For classifiers that output scores or probability estimates, this setting is usually the choice of decision threshold. Area under the curve, shows the global performance of the classifier over a range of possible decision thresholds, where local evaluation metric such as F-measure show the performance for a specific decision threshold. Therefore, when different classifiers are compared in terms of local metrics, the choice of the decision threshold becomes important. The decision threshold may be set to a fixed optimal value with or without considering the operating conditions: the cost proportions or skew levels (Hernández-Orallo et al., 2012). The performance metrics that can be maximized to set the decision threshold are accuracy, Brier score, AUC, expected cost, G-mean and F-measure (Hernández-Orallo et al., 2012; Lipton et al., 2014).

Table 2: Ensembles and their variants.

Abbreviation	Sampling method	Boosting Ensemble	Loss factor	Data
Ada:	Resampling with replacement	AdaBoost Freund et al. (1996)	Weighted accuracy	Synthetic, Video
Ada-F:	Resampling with replacement	Modified AdaBoost Freund et al. (1996)	Proposed F-measure	Synthetic, Video
SMT:	Synthetic minority over-sampling technique (SMOTE)	SMOTEBoost Chawla et al. (2003)	Weighted accuracy	Synthetic, Video
SMT-F	Synthetic minority over-sampling technique(SMOTE)	Modified SMOTEBoost Chawla et al. (2003)	Proposed F-measure	Synthetic, Video
RUS:	Random under-sampling with replacement (RUS)	RUSBoost Seiffert et al. (2010)	Weighted accuracy	Synthetic, Video,
RUS-F:	Random under-sampling with replacement (RUS)	Modified RUSBoost Seiffert et al. (2010)	Proposed F-measure	Synthetic, Video
CUS$_i$:	Selecting ideal clusters generated in synthetic dataset	Growing imbalance Soleymani et al. (2016a)	-	Synthetic
TUS:	Selecting trajectories in video dataset	Growing imbalance Soleymani et al. (2016a)	-	Video
RB:	Combination of up-sampling (SMOTE) and under-sampling (RUS)	RB-Boost Díez-Pastor et al. (2015)	Weighted accuracy	Synthetic, Video
RB-F:	Combination of up-sampling (SMOTE) and under-sampling (RUS)	Modified RB-Boost Díez-Pastor et al. (2015)	Proposed F-measure	Synthetic, Video
PRUS:	Random under-sampling without replacement (RUSwR)	Progressive Boosting	Weighted accuracy	Synthetic, Video
PRUS-F:	Random under-sampling without replacement (RUSwR)	Progressive Boosting	Proposed F-measure	Synthetic, Video
PCUS:	Selecting clusters found by k-means	Progressive Boosting	Weighted accuracy	Synthetic, Video
PCUS-F:	Selecting clusters found by k-means	Progressive Boosting	Proposed F-measure	Synthetic, Video
PCUS$_i$:	Selecting ideal clusters generated in synthetic dataset	Progressive Boosting	Weighted accuracy	Synthetic
PCUS$_i$-F:	Selecting ideal clusters generated in synthetic dataset	Progressive Boosting	Proposed F-measure	Synthetic
PTUS:	Selecting trajectories in video dataset	Progressive Boosting	Weighted accuracy	Video
PTUS-F:	Selecting trajectories in video dataset	Progressive Boosting	Proposed F-measure	Video

To this aim, the classifiers are tested on a set of data called validation datasets that are independent from training and testing data.

When data is imbalanced, the same change in the number of true positives and false positives reflects more significantly in TPR than FPR. Therefore, precision is preferred to FPR because it magnifies FPR by the skew level of data for the given TPR:

$$Pr = \frac{\text{TPR}}{\text{TPR} + \pi\text{FPR}}\ , \tag{35}$$

where π is the proportion of the number of negatives to the positives. Consequently, AUPR (area under precision-recall curve) is preferred to AUC (area under ROC curve) when data is imbalanced. However, positive class is often the class of interest which makes TPR (or recall) very important. In this case, F-measure is a more suitable metric to compare the performance of the classification systems. F_β-measure shows the harmonic mean of precision and recall(Re) when a higher importance is given to recall (in other words changes in precision is eased by β^2).

$$F_\beta = \frac{1}{\frac{(1+\beta^2)^{-1}}{\text{Pr}} + \frac{(1-(1+\beta^2)^{-1})}{\text{Re}}} = \frac{(1+\beta^2)\text{Pr}\cdot\text{Re}}{\beta^2\text{Pr}+\text{Re}} \tag{36}$$

Therefore, in the experiments in this paper, AUPR is used to compare the performance of Boosting ensembles globally which shows the average value of precision for different values of recall (or TPR) giving them equal importance. F_2-measure is used for giving a higher importance to recall and G-mean is used to evaluates the performance of the classification systems giving equal weight to the TPR and TNR$(= 1 - \text{FPR})$.

The values of performance metrics are averaged over 10 replications obtained by 2 $\times$ 5-fold cross validation. In our experiments, the decision threshold to obtain the F_2-measure is set to the value that maximizes the value of F_2-measure on the validation data for comparing the performance of different classification algorithms.

An example is shown in Figure 7, the PR curve of an experiment under setting D_2 on the validation data with skew level of 1:50. In Figure 8, the ensembles are tested on a different test set and F_{op} shows the value of F-measure when the optimal threshold is selected using the validation step described. F_{D} is the value of F-measure when the combination function in Boosting ensembles is majority voting and the decisions of base classifiers are combined. It is observed that F_{op} and F_{D} may differ significantly and in most cases $F_{\text{op}} > F_{\text{D}}$.

In our experiments, the performance of the proposed PBoost ensemble is also compared to state of the art Boosting ensembles in terms of computational complexity. Time complexity for SVM training depends on several factors including the number of training samples, the learning (optimization) algorithm and the number of features. The computational complexity of SVM implemented in LibSVM is evaluated in (Chang & Lin, 2011), as $O(n_{\text{tr}}d)$ per iteration I, where n_{tr} is the training set size, and d is the number of features. The authors state that "the number of iterations p may be higher than linear to the number of training data". Therefore, the complexity is $O(n_{\text{tr}}^p \cdot d)$ for some $p > 2$. This means that, time complexity for SVM training is not proportional to, but increases more than linearly with respect to the training set size.

In the proposed PRUS and baseline Boosting ensembles, p is unknown and d is identical in all algorithms. Each iteration of Boosting ensembles includes a validation step that should be added to training complexity to obtain the overall time complexity of learning process. Time complexity of the validation step $O(n_{\text{SV}} \cdot n_{\text{val}})$, depends on the number of validation samples n_{val} and the number of support vectors n_{SV} obtained from training each SVM. The reason is that, when an RBF SVM with n_{SV} support vectors is tested on a probe sample $\mathbf{x}$, the value of $K(\mathbf{x}, \text{SV}_j) = \exp\{{}^{-\|\mathbf{x}-\text{SV}_j\|^2}/_{2\kappa^2}\}$ is accumulated for all support vectors ($j = 1, \ldots, n_{\text{SV}}$) and the sign of the resulting quantity determines the decision.

Table 3 shows the number of samples to train and validate the ensembles of the size E. The number of validation samples in baseline Boosting ensembles is the same and equal to the overall number of training samples. However, the overall number of

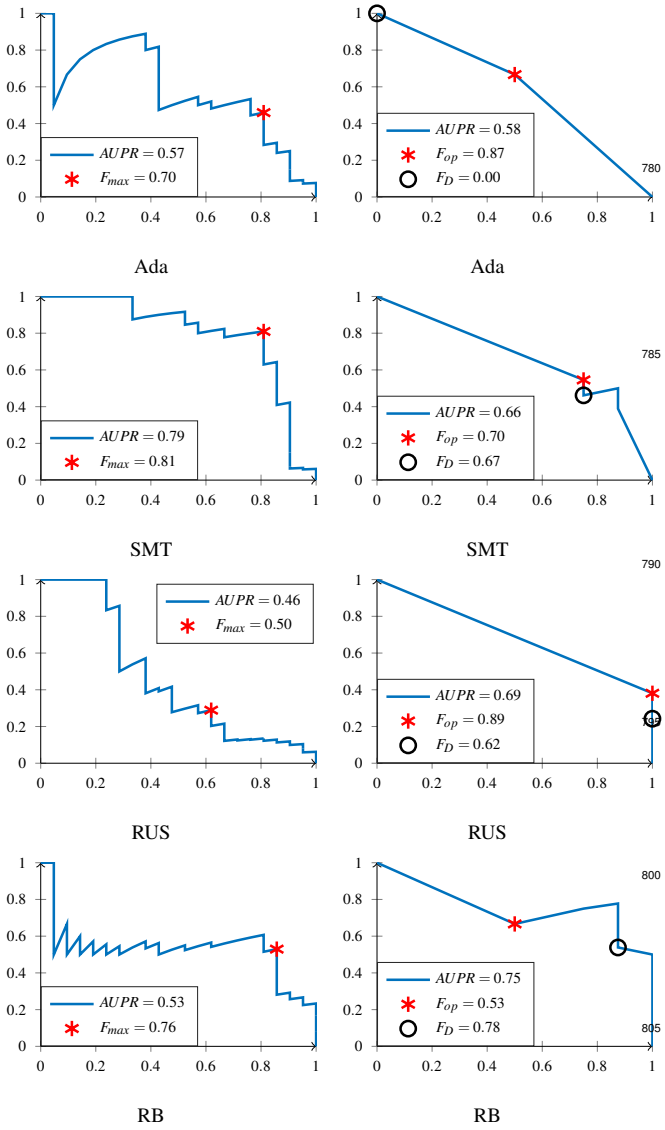


Figure 7: PR curve of baseline Boosting ensembles on validation data and finding the optimal threshold.

Figure 8: PR curve of baseline Boosting ensembles on test data using the optimal threshold obtained from validation step.

samples used for validation in PTUS is calculated as:

$$\sum_{e=1}^{E}(M^+ + \sum_{f=1}^{e} N_f) = EM^+ + \sum_{e=1}^{E}(E-(e-1))N_e, \quad (37)$$

$$= EM^+ + E^2 - \sum_{e=1}^{E} eN_e + M^-, \quad (38)$$

$$= EM^+ + M^- + E^2 - \sum_{e=1}^{E} eN_e. \quad (39)$$

This value is less than $E(M^+ + M^-)$ that is the total number of validation samples in the sate of the art Boosting ensembles. Table 3 shows that the total number of training and validation samples in PTUS ensemble is the smallest one. The number of training and validation samples in TUS is also found using Eq. 39.

5. Results and Discussion

The performance of the proposed and state of the art ensemble learning methods are analyzed for synthetic and video data in 4 parts: (1) accuracy and robustness over different levels of overlap and imbalance between design and test data and of using the proposed loss factor; (2) the performance of RUSBoost with and without progressive partitioning; (3) the combined impact of progressive partitioning and proposed loss factor; (4) the computation complexity during design and testing.

5.1. Results of Experiments with Synthetic Data

5.1.1. Impact of Loss Factor Based on the F-measure

The performance of the baseline Boosting ensembles: AdaBoost , SMOTEBoost, RUSBoost, and RB-Boost are compared in Table 4 for different settings. In addition, F_β is used to optimize loss factor calculation in these ensembles.

Given a fixed skew level of test data, the performance of all Boosting ensembles declines in terms of F-measure and AUPR as the overlap between positive and negative classes grows. This decline of performance is more significant when test data is imbalanced compared to the case where test data is balanced. In our experiments, changes in skew level of test data result in different number of misclassified negative samples and no change in the number of correctly classified positive samples. Therefore, even for the same level of overlap, the performance of all ensembles degrades, in terms of F-measure and AUPR when testing on a more imbalanced data. However, the value of G-mean is relatively high and does not change significantly with variations in imbalance.

For the same level of overlap and different imbalance of training data (settings D_1 and D_3) the performance of all Boosting ensembles is lower when imbalance of training data is lower. The reason is that less information is provided for training and also the skew level of training and test data has a greater difference. Overall, Table 4 shows that SMT and RB are the most robust to changes in overlap and imbalance. Based on the Table 4, the following results are obtained. Using the F-measure loss factor improves the performance of RUS significantly with D_1, D_2 and D_3 for all λ_{test} in terms of all three metrics. Performance of SMT and RB improves only in terms of AUPR and with D_1 and D_2. Performance of Ada improves when $\lambda_{\text{test}} = 1:100$ in terms of G-mean and AUPR.

Using the F-measure loss factor may improve the performance of the Boosting ensembles that rely on under-sampling of data in terms of F-measure, especially for more difficult problems with overlapping data. The performance of Boosting ensembles that involve up-sampling of positive samples does not improve significantly in terms of F-measure. However, the global performance of these Boosting ensembles in terms of AUPR does improve after using the F-measure loss factor.

Table 3: Number of training and validation samples.

Ensemble	n_{tr} in iteration e	Total n_{tr}	n_{val} in iteration e	Total n_{val}
Ada	$M^+ + M^-$	$E(M^+ + M^-)$	$M^+ + M^-$	$E(M^+ + M^-)$
SMT	$2M^-$	$2EM^-$	$M^+ + M^-$	$E(M^+ + M^-)$
RUS	$2M^+$	$2EM^+$	$M^+ + M^-$	$E(M^+ + M^-)$
TUS	$M^+ + \sum_{f=1}^{e} N_f$	$EM^+ + M^- + E^2 - \sum_{e=1}^{E} eN_e$	$M^+ + \sum_{f=1}^{e} N_f$	$EM^+ + M^- + E^2 - \sum_{e=1}^{E} eN_e$
RB	$M^+ + M^-$	$E(M^+ + M^-)$	$M^+ + M^-$	$E(M^+ + M^-)$
PTUS	$M^+ + N_e$	$EM^+ + M^-$	$M^+ + \sum_{f=1}^{e} N_f$	$EM^+ + M^- + E^2 - \sum_{e=1}^{E} eN_e$

In Table 5, the performance of baseline ensembles and their variants for different values of β is compared for $D2$. The goal is to evaluate the effect of the value of β on improving the performance when the proposed loss factor is used. This Table shows the performance only when $\lambda_{\text{test}} = 1:100$ because the performance of baseline systems usually decline for higher skew levels of test data.

Evaluation is done in terms of the same F_β-measure that is used in loss factor calculation. The results are shown in terms of both F_D and F_{op}. F_D is the value of F-measure when the decisions of base classifiers are combined in Boosting ensembles and F_{op} is the value of F-measure when the scores of base classifiers are combined in Boosting ensembles and the optimal decision threshold of each ensemble is set to the point that maximizes F-measure when that ensemble is validated on an independent set of data (see section 4.2.1.).

The performance of Ada improves for all values of β in terms of both F_D and F_{op}. Some improvements are seen for SMT and RB, but F_D and F_{op} tend to stay the same in most cases and decrease in a few cases. The performance of RUS improves for $\beta = 1$ and 2 in terms of both F_D and F_{op}, and the improvement tends to decrease for higher β values. This was expected, since using higher values of β to calculate F-measure means giving more importance to recall than precision. Therefore, the impact of imbalance is masked when higher values of β is used and the performance may not change when the loss factor is calculated based on F-measure. For each of the classification systems, the same reason result in higher values of F_D and F_{op} with higher values of β. Comparing F_D and F_{op} of each ensemble for each value of β shows that selecting the proper decision threshold can improve the performance in terms of accuracy and robustness, especially for lower values of β.

The results are not shown in terms of AUPR because AUPR does not change with variations in the value of β, since the importance of recall and precision stays the same and equal in obtaining AUPR.

5.1.2. *Impact of progressive partitioning in RUSBoost*

In this section, progressive partitioning is integrated into RUS without the use of F-measure in loss factor calculation. It is observed in Table 7 that robustness of RUS improves significantly after using this method of sampling. Indeed, using all samples for training through partitioning avoids loss of information and may improve the classification accuracy. In addition, validating on different imbalance levels of data increases the robustness to variations in the imbalance level of test data. The performance of PCUS_i is significantly better than RUS in terms of both F-measure and G-mean with D_1, D_2 and D_3 and all λ_{test}. In terms of AUPR, integrating PCUS_i improves the performance of RUS with D_2, and higher skew levels of test data with D_1 and D_2. Integrating PRUS and PCUS also improves the performance of RUS specially in terms of F-measure and AUPR and with D_1 and D_3.

5.1.3. *Impact of progressive partitioning and loss factor combined*

In this section progressive partitioning and the proposed loss factor are integrated into RUS algorithm, resulting in PRUS-F, PCUS-F, and PCUS_i-F. In terms of F-measure, PCUS_i-F outperforms other classification systems for higher skew levels of test data. PCUS_i-F outperforms others in terms of G-mean with D_2. Comparing Tables 4, 6 and 7 shows that combining the use of F-measure and progressive partitioning is more effective in increasing performance and robustness compared to using each of them independently because during learning process, accuracy and robustness to imbalance improve at the same time, not separately. If the negative class is partitioned a priori (CUS_i), PBoost performs significantly better than the case when general partitioning techniques (RUS and CUS) are used.

5.2. *Results of Experiments with Video Data*

Similarly to the synthetic data sets, the results of experiments on video datasets are shown in three parts, assessing the impact of: (1) using the F-measure loss factor on the performance of baseline Boosting ensembles, (2) integrating progressive partitioning into RUS, and (3) using the F-measure loss factor and progressive partitioning compared with the baseline and state of the art ensembles.

From Table 8, the performance level of all ensembles is lower when the skew level of training data is higher. This is despite the fact that when the imbalance of training data is lower, the data that is used to test classifiers contain samples from some individuals that are not in the training data. Using the F-measure loss factor improves the performance of Ada and RUS in terms of F-measure, and has less impact on the performance of RB and SMT in most cases of skew between classes in training and testing data. In terms of G-mean, the performance improves for all ensembles with both datasets, except for RUS with FIA. The performance of these ensembles after using the F-measure loss factor does not change significantly in terms of AUPR.In

fact, the use of F-measure loss factor performs similarly to adjusting the decision threshold of the Boosting algorithms to better account for imbalance and therefore may improve the performance only in terms of local performance metrics like F-measure.

After integrating the progressive partitioning in RUS using PRUS and PTUS, the performance of RUS improves and becomes more robust in terms of both F-measure and AUPR (see Table 9), especially when TUS is used for partitioning. Validating base classifiers on different imbalance levels of imbalance result in more robust classification systems and using all samples for training through partitioning avoids loss of information and may improve the classification accuracy.

Comparing the performance of final PBoost variants with baseline ensembles in Table 10, PTUS-F outperforms all other approaches in terms of F-measure. In terms of G-mean RUS performs the best and in terms of AUPR, SMT has the highest mean value. From these results, it is observed that combining the use of F-measure and integration of progressive partitioning, is more effective in increasing performance and robustness compared to using each of them independently. In the experiments on the video data, trajectory under-sampling is more effective when used in PBoost compared to random under-sampling without replacement and cluster under-sampling. This is the case when partitions of negative class are known a priori.

5.3. Statistical Comparison of the Classification Systems

The statistical comparison in this paper is carried out once for each skew level of test data. $K = 8$ is the number of classification systems in Tables 7 and 10. $N = 8$ is the number of independent datasets used for statistical comparison that includes six synthetic datasets and two video datasets. The results of experiments with three of the synthetic datasets D_1, D_2 and D_3 have been presented in section 5.1. Remaining three datasets used for statistical comparison are $D_4(\lambda_{\text{train}} = 1:20, \delta = 0.1)$, $D_5(\lambda_{\text{train}} = 1:100, \delta = 0.2)$, $D_6(\lambda_{\text{train}} = 1:100, \delta = 0.1)$. The data used for two settings of experiments on video data are dependent because all the samples that are used when $\lambda_{\text{train}} = 1:50$ are used again when $\lambda_{\text{train}} = 1:100$. Therefore, only the results with $\lambda_{\text{train}} = 1:100$ are used for statistical comparison of the classification systems with FIA and COX datasets.

For each skew level of test data (s=1,...,4), Table 11 shows the mean rank of each classification system ($R_k^s = (1/N)\sum_n r_{\text{n,s}}^k$ where $k = 1,\ldots,K$ and $n = 1,\ldots,N$). $r_{\text{n,s}}^k$ is the rank of k^{th} classification system for s^{th} skew level with n^{th} dataset such that best performing algorithm gets rank of 1 and the worst one gets rank of 8. In case of ties average ranks are assigned. From Table 11, it is observed that PTUS-F takes the highest rank in terms of F-measure and G-mean. To determine if there are any significant differences between the ranks, we use Friedman test and the subsequent version of Iman and Davenport (Iman & Davenport, 1980) which rejects the null hypothesis that all classification systems perform the same. Then we use Hochberg test (Hochberg, 1988) to compare the state of the art ensemble methods (Ada, RUS, SMT, RB and TUS) to PTUS-F. Table 12 shows the p-values of these comparisons. With alpha of 0.05, the boldface entries of the table indicate the rejection of null hypothesis that PTUS-F does not outperform these ensembles. It is observed that the null hypothesis is rejected in 14 out of 20 comparisons in terms of F-measure, and 6 out of 20 comparisons in terms of G-mean and AUPR.

5.4. Computational Complexity

In this section the time complexity needed to design and test the proposed and baseline Boosting ensembles are compared. To compare the training time and memory cost of these ensembles, the number of training samples is counted and to compare their validation time and memory cost the number of validation samples and the number of support vectors of base classifiers are considered.

Figure 9 show the results obtained with setting D_2 in our experiments. The number of training and validation samples, the average number of support vectors, and overall number of evaluations of the kernel function ($n_{\text{SV}} \cdot n_{\text{val}}$) is presented in Figure 9(a)-(d) to estimate and compare design time of the proposed and baseline Boosting ensembles. To compare the complexity of these classification systems during testing $O(n_{\text{SV}})$ with a probe sample $\mathbf{x}$, we compared the overall number of support vectors in these ensembles in Figure 9(e) because computing each SVM output requires n_{SV} evaluations of the kernel function.

Given n_{tr}^e as the number of samples to train the e^{th} classifier in the ensemble, Figure 9(a) shows $\sum_{\text{e=1}}^{\text{E}} n_{\text{tr}}^e$. In Figure 9(b), $\sum_{\text{e=1}}^{\text{E}} n_{\text{val}}^e$ is presented, where n_{val}^e is the number of samples that the e^{th} classifier in the ensemble is validated with. Average number of support vectors in E classifiers of the ensembles are shown in Figure 9(c). Given n_{SV}^e as the number of support vectors obtained after training the e^{th} classifier in the ensemble, Figure 9(d) shows $\sum_{\text{e=1}}^{\text{E}} n_{\text{val}}^e \cdot n_{\text{SV}}^e$ for each ensemble.

In terms of training (see Figure 9(a)), PCUS$_i$ and RUS are under-sampling ensembles and have the lowest computational cost, while SMT and RB-Boost include up-sampling and are significantly more costly. Total number of validation samples is equal for Ada, SMT, RUS and RB, and total number of validation samples is less with PCUS$_i$ (see Figure 9(b)). The average number of support vectors is higher for SMT (see Figure 9(c)) because the base classifiers in this ensemble are trained on higher number of samples. Therefore, SMT is the most costly method, in terms of validation (see Figure 9(d)). Note that time and memory required for partitioning in PCUS$_i$, and generating synthetic samples in SMT and RB-Boost is neglected here. Nevertheless, PCUS$_i$ is the most efficient ensemble technique in terms of designing memory and time complexity.

The number of training and validation samples as well as the average number of support vectors is smaller with PCUS$_i$ and therefore, PCUS$_i$ is less costly in terms of design time and memory complexity.

In terms of testing time complexity (see Figure 9(e)) PCUS$_i$ and RUS have the lowest number of evaluations of the kernel function per probe sample. RB-Boost and SMT have the highest number of evaluations of the kernel function per probe sample.

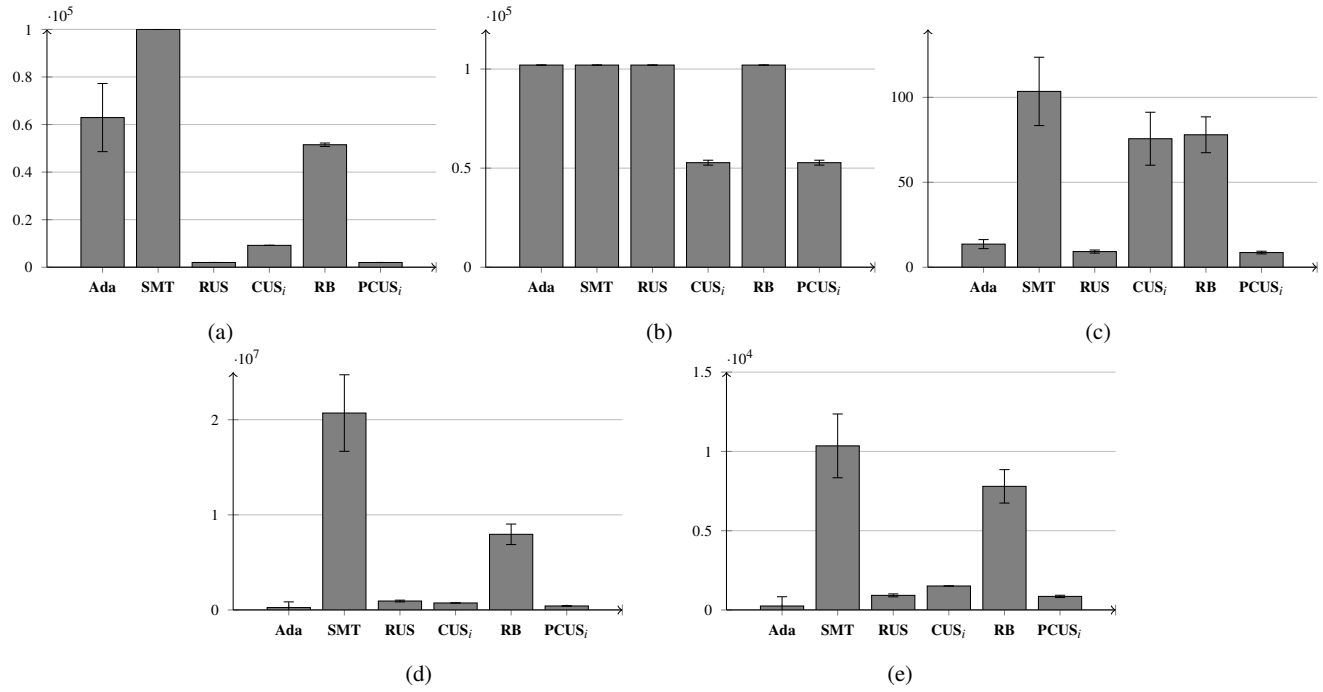


Figure 9: Complexity related to the design and testing process of ensembles: (a) total number of training samples, (b) total number of validation samples, (c) Number of $n_{\text{SV}} \cdot n_{\text{val}}$ during validation, (d) total number of $n_{\text{SV}} \cdot n_{\text{val}}$ during validation, (e) total number of evaluations of the kernel function per probe sample during testing.

Although AdaBoost is given the same ensemble size, it fails to generate enough classifiers [2] and consequently result in smaller number of support vectors. Therefore, the total number of validation and testing processes of Ada is lower than expected.

5.5. Summary of Results

As a summary of results on synthetic and video datasets, we observed that:

1. Using the proposed loss factor calculation may reduce the bias of performance in Boosting ensembles and increase the accuracy.

2. Partitioning improves the performance of RUS in all cases in terms of both accuracy and robustness to imbalance.

3. Integrating both partitioning and the proposed loss factor outperforms state of the art Boosting ensembles, relying on the choice of partitioning technique for each dataset such that:

   (a) With synthetic data, PCUS$_i$-F outperforms all systems in terms of both F-measure and AUPR, while PRUS and PCUS outperform RUS in most cases of skew and overlap between classes.

   (b) With the video data, PTUS is more accurate than the state of the art ensembles, PRUS as well as PCUS.

4. PBoost is computationally less costly than the state of the art Boosting ensembles in terms of computational complexity.

Therefore, PBoost is an effective approach in correct classification of data when data is imbalanced in comparison to the state of the art Boosting ensembles especially for face re-identification. This method relies on the choice of partitioning technique for each dataset and performs significantly better when a more suitable partitioning technique is used. In problems that the natural clusters are known, as with trajectory under-sampling in face re-identification application, the performance is better than using the general partitioning methods such as random under-sampling without replacement or k-means clustering. Therefore, PBoost can be more efficient than baseline Boosting ensembles for face re-identification under imbalance considering both accuracy and complexity factors.

6. Conclusion

In this paper, a new Boosting ensemble algorithm named as PBoost is proposed to address imbalance based on the idea of modifying RUSBoost by (1) under-sampling the majority class

[2] AdaBoost failed many times to generate classifiers because the training subset in step 2.i of Algo. 1 may contain only negative class samples during sampling or weight update in step 2.vii may lead to that due to unsuitable loss factor calculation in step 2.ivv. Nevertheless, only successful attempts are considered.

using partitional techniques and in particular trajectory-based partitioning, inspired by face re-identification applications, (2) validating classifiers on a growing validation subset, and (3) using a more suitable loss factor calculation. The partitions enter the Boosting process progressively for designing classifiers over iterations to avoid information loss and to maintain diversity among them. Validating base classifiers on a growing number of negative samples makes the PBoost ensembles more robust to possible skew levels of data during operations in addition to lowering the computational complexity. The loss factor based on F-measure handles bias of performance towards negative class, and guides the Boosting process in a more effective direction with the purpose of correctly classifying both classes. Experiments show that PBoost may perform differently with different techniques of partitioning for each dataset such that more suitable partitioning result in better performance. For face re-identification application, the PBoost ensemble using trajectory under-sampling outperforms the state of the art Boosting ensembles in terms of accuracy. In addition, PBoost has significantly lower computational complexity in both designing and testing stages compared to state of the art ensembles. The applicability of PBoost to multi-class classification problems and other real-world applications can be further investigated in a future work.

Acknowledgement

This work was partially supported by the Natural Sciences and Engineering Research Council of Canada and Mitacs. This work has been also partially supported by the LETSCROWD project, funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 740466.

References

Branco, P., Torgo, L., & Ribeiro, R. P. (2016). A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, *49*, 31.

Chang, C.-C., & Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*, 27.

Chawla, N. V., Lazarevic, A., Hall, L. O., & Bowyer, K. W. (2003). Smoteboost: Improving prediction of the minority class in boosting. In *European Conference on Principles of Data Mining and Knowledge Discovery* (pp. 107–119). Springer.

Díez-Pastor, J. F., Rodríguez, J. J., García-Osorio, C., & Kuncheva, L. I. (2015). Random balance: Ensembles of variable priors classifiers for imbalanced data. *Knowledge-Based Systems*, *85*, 96–111.

Dunn, J. C. (1973). A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters, .

Fan, W., Stolfo, S. J., Zhang, J., & Chan, P. K. (1999). Adacost: misclassification cost-sensitive boosting. In *Machine Learning and Applications, 1999. ICMLA'99 International Conference on.*

Flach, P., & Kull, M. (2015). Precision-recall-gain curves: Pr analysis done right. In *Advances in Neural Information Processing Systems* (pp. 838–846).

Freund, Y., & Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory* (pp. 23–37). Springer.

Freund, Y., Schapire, R. E. et al. (1996). Experiments with a new boosting algorithm. In *Machine Learning and Applications, 1996. ICMLA'96 International Conference on.*

Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, *44*, 1761–1776.

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. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *42*, 463–484.

Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2013a). Dynamic classifier selection for one-vs-one strategy: avoiding non-competent classifiers. *Pattern Recognition*, *46*, 3412–3424.

Galar, M., Fernández, A., Barrenechea, E., & Herrera, F. (2013b). Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, *46*, 3460–3471.

García, S., & Herrera, F. (2009). Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary computation*, *17*, 275–306.

Goh, R., Liu, L., Liu, X., & Chen, T. (2005). The cmu face in action (fia) database. In *Analysis and Modelling of Faces and Gestures* (pp. 255–263).

Guo, H., & Viktor, H. L. (2004). Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *ACM SIGKDD Explorations Newsletter*, *6*, 30–39.

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2016). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, .

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, *21*, 1263–1284.

Hernández-Orallo, J., Flach, P., & Ferri, C. (2012). A unified view of performance metrics: translating threshold choice into expected classification loss. *Journal of Machine Learning Research*, *13*, 2813–2869.

Hochberg, Y. (1988). A sharper bonferroni procedure for multiple tests of significance. *Biometrika*, *75*, 800–802.

Hu, S., Liang, Y., Ma, L., & He, Y. (2009). Msmote: improving classification performance when training data is imbalanced. In *Proceedings of the 2009 Second International Workshop on Computer Science and Engineering* (pp. 13–17). Citeseer volume 2.

Huang, Z., Shan, S., Wang, R., Zhang, H., Lao, S., Kuerban, A., & Chen, X. (2015). A benchmark and comparative study of video-based face recognition on cox face database. *IEEE Transactions on Image Processing*, *24*, 5967–5981.

Iman, R. L., & Davenport, J. M. (1980). Approximations of the critical region of the fbietkan statistic. *Communications in Statistics-Theory and Methods*, *9*, 571–595.

Joshi, M. V., Kumar, V., & Agarwal, R. C. (2001). Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on* (pp. 257–264). IEEE.

Kim, M.-J., Kang, D.-K., & Kim, H. B. (2015). Geometric mean based boosting algorithm with over-sampling to resolve data imbalance problem for bankruptcy prediction. *Expert Systems with Applications*, *42*, 1074–1082.

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, *5*, 221–232.

Li, Q., Yang, B., Li, Y., Deng, N., & Jing, L. (2013). Constructing support vector machine ensemble with segmentation for imbalanced datasets. *Neural Computing and Applications*, *22*, 249–256.

Li, X., Wang, L., & Sung, E. (2008). Adaboost with svm-based component classifiers. *Engineering Applications of Artificial Intelligence*, *21*, 785–795.

Lipton, Z. C., Elkan, C., & Naryanaswamy, B. (2014). Optimal thresholding of classifiers to maximize f1 measure. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 225–239). Springer.

López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, *250*, 113–141.

Mease, D., Wyner, A., & Buja, A. (2007). Cost-weighted boosting with jittering and over/under-sampling: Jous-boost. *J. Machine Learning Research*, *8*, 409–439.

Ojala, T., Pietikainen, M., & Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *24*, 971–987.

Radtke, P. V., Granger, E., Sabourin, R., & Gorodnichy, D. O. (2014). Skew-sensitive boolean combination for adaptive ensembles–an application to face recognition in video surveillance. *Information Fusion*, *20*, 31–48.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, *33*, 1–39.

Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, *100*, 401–409.

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2010). Rusboost: A hybrid approach to alleviating class imbalance. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *40*, 185–197.

Soleymani, R., Granger, E., & Fumera, G. (2016a). Classifier ensembles with trajectory under-sampling for face re-identification. In *Proceedings of the International Conference on Pattern Recognition Applications and Methods-Volume 1* (pp. 97–108). SCITEPRESS-Science and Technology Publications, Lda.

Soleymani, R., Granger, E., & Fumera, G. (2016b). Loss factors for learning boosting ensembles from imbalanced data. In *Pattern Recognition (ICPR), 2016 23rd International Conference on* (pp. 204–209). IEEE.

Sun, Y., Kamel, M. S., Wong, A. K., & Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, *40*, 3358–3378.

Ting, K. M. (2000). A comparative study of cost-sensitive boosting algorithms. In *Machine Learning and Applications, 2000. International Conference on*. Citeseer.

De-la Torre, M., Granger, E., Sabourin, R., & Gorodnichy, D. O. (2015). Adaptive skew-sensitive ensembles for face recognition in video surveillance. *Pattern Recognition*, *48*, 3385–3406.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (pp. I–511). IEEE volume 1.

Wang, S., & Yao, X. (2012). Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *42*, 1119–1130.

Xiao, J., Xie, L., He, C., & Jiang, X. (2012). Dynamic classifier ensemble model for customer classification with imbalanced class distribution. *Expert Systems with Applications*, *39*, 3668–3675.

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, *16*, 645–678.

Yan, R., Liu, Y., Jin, R., & Hauptmann, A. (2003). On predicting rare classes with svm ensembles in scene classification. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on* (pp. III–21). IEEE volume 3.

Table 4: Average of F_2-measure, G-mean and AUPR performance of baseline techniques with and without F-measure loss factor on synthetic data over different levels of skew and overlap in test data.

Ensembles	Train Data	D_1 (λ_{train} =1:50, $\delta = 0.2$)				D_2 (λ_{train} =1:50, $\delta = 0.1$)				D_3 (λ_{train} =1:20, $\delta = 0.2$)			
	λ_{test}	**1:1**	**1:20**	**1:50**	**1:100**	**1:1**	**1:20**	**1:50**	**1:100**	**1:1**	**1:20**	**1:50**	**1:100**
F_2-measure													
Ada		0.97 ± 0.02	0.97 ± 0.03	0.92 ± 0.02	0.78 ± 0.04	0.85 ± 0.05	0.75 ± 0.07	0.62 ± 0.05	0.42 ± 0.05	0.98 ± 0.02	0.93 ± 0.03	0.85 ± 0.02	0.57 ± 0.04
Ada-F		0.96 ± 0.03	0.96 ± 0.03	0.92 ± 0.03	**0.79** ± 0.04	**0.87** ± 0.09	0.74 ± 0.08	0.58 ± 0.05	0.42 ± 0.08	0.98 ± 0.02	0.92 ± 0.03	0.85 ± 0.02	**0.58** ± 0.05
RUS		0.89 ± 0.11	0.85 ± 0.21	0.55 ± 0.19	0.34 ± 0.15	0.82 ± 0.09	0.62 ± 0.10	0.47 ± 0.08	0.27 ± 0.08	0.56 ± 0.20	0.35 ± 0.24	0.21 ± 0.18	0.13 ± 0.11
RUS-F		**0.93** ± 0.06	**0.93** ± 0.06	**0.81** ± 0.11	**0.63** ± 0.17	**0.93** ± 0.04	**0.71** ± 0.06	**0.51** ± 0.03	**0.36** ± 0.07	**0.85** ± 0.15	**0.77** ± 0.20	**0.67** ± 0.21	**0.47** ± 0.18
SMT		0.96 ± 0.04	0.96 ± 0.04	0.94 ± 0.04	0.90 ± 0.03	0.94 ± 0.04	0.85 ± 0.02	0.65 ± 0.02	0.61 ± 0.02	0.96 ± 0.04	0.91 ± 0.01	0.90 ± 0.02	0.74 ± 0.03
SMT-F		0.95 ± 0.04	0.95 ± 0.04	0.94 ± 0.04	0.90 ± 0.03	0.94 ± 0.04	0.85 ± 0.02	0.65 ± 0.02	0.61 ± 0.02	0.96 ± 0.04	0.91 ± 0.01	0.90 ± 0.02	0.74 ± 0.03
RB		0.96 ± 0.02	0.96 ± 0.02	0.94 ± 0.02	0.90 ± 0.01	0.91 ± 0.05	0.85 ± 0.01	0.63 ± 0.01	0.60 ± 0.01	0.96 ± 0.03	0.92 ± 0.02	0.90 ± 0.02	0.73 ± 0.05
RB-F		0.96 ± 0.03	0.96 ± 0.03	0.94 ± 0.03	0.90 ± 0.02	0.91 ± 0.03	0.85 ± 0.02	0.63 ± 0.02	0.59 ± 0.03	0.97 ± 0.03	0.91 ± 0.01	0.89 ± 0.03	0.72 ± 0.05
G-mean													
Ada		0.97 ± 0.02	0.97 ± 0.02	0.97 ± 0.02	0.97 ± 0.02	0.86 ± 0.05	0.86 ± 0.05	0.86 ± 0.05	0.85 ± 0.05	0.98 ± 0.02	0.98 ± 0.02	0.98 ± 0.02	0.97 ± 0.02
Ada-F		0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03	**0.88** ± 0.08	**0.87** ± 0.07	**0.87** ± 0.07	**0.87** ± 0.07	0.98 ± 0.02	0.98 ± 0.02	0.98 ± 0.02	0.97 ± 0.02
RUS		0.93 ± 0.05	0.93 ± 0.05	0.92 ± 0.05	0.91 ± 0.06	0.84 ± 0.08	0.83 ± 0.08	0.83 ± 0.08	0.82 ± 0.08	0.50 ± 0.20	0.68 ± 0.23	0.68 ± 0.23	0.68 ± 0.23
RUS-F		**0.94** ± 0.06	**0.94** ± 0.06	**0.93** ± 0.06	**0.93** ± 0.05	**0.93** ± 0.04	**0.92** ± 0.04	**0.92** ± 0.04	**0.92** ± 0.04	**0.80** ± 0.25	**0.91** ± 0.08	**0.91** ± 0.08	**0.91** ± 0.08
SMT		0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03	0.94 ± 0.04	0.94 ± 0.04	0.94 ± 0.04	0.94 ± 0.04	0.96 ± 0.04	0.96 ± 0.03	0.96 ± 0.04	0.96 ± 0.04
SMT-F		0.96 ± 0.04	0.96 ± 0.04	0.96 ± 0.04	0.95 ± 0.04	0.94 ± 0.03	0.94 ± 0.03	0.93 ± 0.03	0.94 ± 0.03	0.96 ± 0.04	0.96 ± 0.03	0.96 ± 0.04	0.96 ± 0.04
RB		0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.02	0.91 ± 0.05	0.91 ± 0.04	0.90 ± 0.04	0.91 ± 0.05	0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03
RB-F		0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03	0.91 ± 0.03	0.91 ± 0.03	0.90 ± 0.03	0.91 ± 0.03	0.97 ± 0.03	0.96 ± 0.03	0.97 ± 0.03	0.96 ± 0.03
AUPR													
Ada		1.00 ± 0.00	1.00 ± 0.00	0.98 ± 0.01	0.89 ± 0.04	1.00 ± 0.00	0.82 ± 0.07	0.66 ± 0.05	0.34 ± 0.07	1.00 ± 0.00	0.99 ± 0.01	0.96 ± 0.01	0.70 ± 0.04
Ada-F		1.00 ± 0.00	1.00 ± 0.00	0.98 ± 0.01	**0.90** ± 0.03	1.00 ± 0.00	0.81 ± 0.08	0.55 ± 0.13	**0.36** ± 0.14	1.00 ± 0.00	0.99 ± 0.01	**0.95** ± 0.02	**0.71** ± 0.05
RUS		0.93 ± 0.21	0.90 ± 0.28	0.46 ± 0.25	0.29 ± 0.23	1.00 ± 0.00	0.70 ± 0.13	0.40 ± 0.08	0.20 ± 0.08	0.64 ± 0.20	0.34 ± 0.33	0.20 ± 0.23	0.10 ± 0.11
RUS-F		**1.00** ± 0.00	**1.00** ± 0.00	**0.86** ± 0.19	**0.71** ± 0.23	**1.00** ± 0.00	**0.75** ± 0.11	**0.37** ± 0.07	**0.26** ± 0.09	**0.89** ± 0.22	**0.81** ± 0.27	**0.73** ± 0.30	**0.51** ± 0.27
SMT		0.90 ± 0.32	0.90 ± 0.32	0.89 ± 0.31	0.88 ± 0.31	0.90 ± 0.32	0.84 ± 0.30	0.52 ± 0.19	0.51 ± 0.19	1.00 ± 0.00	0.99 ± 0.01	0.98 ± 0.01	0.88 ± 0.01
SMT-F		**1.00** ± 0.00	**1.00** ± 0.00	**0.99** ± 0.01	**0.98** ± 0.01	**1.00** ± 0.00	**0.93** ± 0.03	**0.58** ± 0.04	**0.57** ± 0.05	1.00 ± 0.00	0.99 ± 0.01	0.98 ± 0.01	0.88 ± 0.01
RB		0.80 ± 0.42	0.80 ± 0.42	0.79 ± 0.42	0.78 ± 0.41	0.60 ± 0.52	0.57 ± 0.49	0.32 ± 0.28	0.31 ± 0.27	1.00 ± 0.00	0.99 ± 0.01	0.98 ± 0.01	0.84 ± 0.03
RB-F		**1.00** ± 0.00	**1.00** ± 0.00	**0.99** ± 0.00	**0.98** ± 0.01	**1.00** ± 0.00	**0.94** ± 0.02	**0.53** ± 0.03	**0.51** ± 0.03	1.00 ± 0.00	0.98 ± 0.01	0.98 ± 0.01	**0.85** ± 0.02

The boldface entries correspond to improvement in the performance of the Boosting ensemble after modifying loss factor using F-measure.

Table 5: Average of F_2-measure performance of baseline techniques with and without F-measure loss factor for different values of β on D_2, $\lambda_{test} = 1:100$.

Ensembles	F_D					F_{op}				
	$\beta = 1$	$\beta = 2$	$\beta = 4$	$\beta = 7$	$\beta = 10$	$\beta = 1$	$\beta = 2$	$\beta = 4$	$\beta = 7$	$\beta = 10$
Ada	0.24 ± 0.14	0.26 ± 0.15	0.19 ± 0.17	0.20 ± 0.15	0.24 ± 0.16	0.38 ± 0.10	0.25 ± 0.18	0.27 ± 0.16	0.34 ± 0.06	0.32 ± 0.13
Ada-F	**0.30** ± 0.07	**0.31** ± 0.07	**0.33** ± 0.05	**0.34** ± 0.03	**0.33** ± 0.04	**0.41** ± 0.09	**0.39** ± 0.11	**0.36** ± 0.05	**0.36** ± 0.08	**0.38** ± 0.07
RUS	0.09 ± 0.01	0.09 ± 0.01	0.09 ± 0.01	0.09 ± 0.01	0.09 ± 0.01	0.51 ± 0.03	0.46 ± 0.03	0.48 ± 0.04	0.43 ± 0.13	0.48 ± 0.03
RUS-F	**0.19** ± 0.04	**0.10** ± 0.01	0.09 ± 0.01	0.09 ± 0.01	0.08 ± 0.02	**0.52** ± 0.08	**0.48** ± 0.03	0.43 ± 0.10	**0.44** ± 0.12	0.46 ± 0.07
SMT	0.36 ± 0.02	0.36 ± 0.02	0.36 ± 0.02	0.33 ± 0.12	0.33 ± 0.12	0.49 ± 0.05	0.47 ± 0.04	0.41 ± 0.15	0.41 ± 0.15	0.45 ± 0.03
SMT-F	0.36 ± 0.02	0.36 ± 0.02	0.36 ± 0.02	**0.37** ± 0.02	**0.36** ± 0.02	0.49 ± 0.05	0.47 ± 0.04	**0.45** ± 0.03	**0.45** ± 0.03	0.45 ± 0.03
RB	0.33 ± 0.01	0.32 ± 0.02	0.29 ± 0.10	0.26 ± 0.14	0.23 ± 0.16	0.47 ± 0.03	0.41 ± 0.15	0.34 ± 0.18	0.38 ± 0.14	0.29 ± 0.20
RB-F	0.30 ± 0.02	0.30 ± 0.02	**0.30** ± 0.02	**0.30** ± 0.02	**0.31** ± 0.02	0.47 ± 0.04	**0.46** ± 0.05	**0.41** ± 0.02	**0.42** ± 0.02	**0.41** ± 0.02

The boldface entries correspond to improvement in the performance of the Boosting ensemble after modifying loss factor using F-measure.

Table 6: Average of F_2-measure, G-mean and AUPR performance of RUSBoost with and without integrating progressive Boosting on synthetic data over different levels of skew and overlap of test data.

Ensembles	**Train Data**	D_1 (λ_{train} =1:50, $\delta = 0.2$)				D_2 (λ_{train} =1:50, $\delta = 0.1$)				D_3 (λ_{train} =1:20, $\delta = 0.2$)			
	λ_{test}	**1:1**	**1:20**	**1:50**	**1:100**	**1:1**	**1:20**	**1:50**	**1:100**	**1:1**	**1:20**	**1:50**	**1:100**
F_2-measure													
RUS		0.89 ± 0.11	0.85 ± 0.21	0.55 ± 0.19	0.34 ± 0.15	0.82 ± 0.09	0.62 ± 0.10	0.47 ± 0.08	0.27 ± 0.08	0.56 ± 0.20	0.35 ± 0.24	0.21 ± 0.18	0.13 ± 0.11
PRUS		0.82 ± 0.09	0.64 ± 0.11	**0.61** ± 0.12	**0.55** ± 0.15	**0.90** ± 0.09	0.43 ± 0.18	0.35 ± 0.14	**0.32** ± 0.11	**0.70** ± 0.06	**0.58** ± 0.18	**0.56** ± 0.20	**0.48** ± 0.19
PCUS		0.83 ± 0.12	0.61 ± 0.17	**0.58** ± 0.16	**0.54** ± 0.14	**0.92** ± 0.13	0.60 ± 0.15	**0.48** ± 0.09	**0.43** ± 0.09	**0.88** ± 0.09	**0.84** ± 0.09	**0.82** ± 0.08	**0.71** ± 0.06
$PCUS_i$		**0.97** ± 0.03	**0.92** ± 0.03	**0.92** ± 0.03	**0.90** ± 0.03	**0.99** ± 0.01	**0.79** ± 0.09	**0.63** ± 0.09	**0.58** ± 0.12	**0.94** ± 0.05	**0.92** ± 0.05	**0.91** ± 0.04	**0.78** ± 0.04
G-mean													
RUS		0.93 ± 0.05	0.93 ± 0.05	0.92 ± 0.05	0.91 ± 0.06	0.84 ± 0.08	0.83 ± 0.08	0.83 ± 0.08	0.82 ± 0.08	0.50 ± 0.20	0.68 ± 0.23	0.68 ± 0.23	0.68 ± 0.23
PRUS		0.85 ± 0.06	0.70 ± 0.09	0.69 ± 0.09	0.69 ± 0.09	**0.91** ± 0.08	0.57 ± 0.15	0.53 ± 0.17	0.53 ± 0.17	**0.78** ± 0.05	**0.76** ± 0.06	**0.76** ± 0.06	**0.76** ± 0.06
PCUS		0.86 ± 0.11	0.69 ± 0.14	0.69 ± 0.14	0.67 ± 0.15	**0.95** ± 0.08	0.71 ± 0.15	0.70 ± 0.15	0.69 ± 0.16	**0.89** ± 0.08	**0.87** ± 0.07	**0.87** ± 0.08	**0.85** ± 0.07
$PCUS_i$		**0.97** ± 0.03	**0.93** ± 0.03	**0.93** ± 0.03	**0.93** ± 0.03	**0.99** ± 0.01	**0.94** ± 0.02	**0.93** ± 0.02	**0.93** ± 0.06	**0.94** ± 0.05	**0.94** ± 0.05	**0.93** ± 0.04	**0.92** ± 0.05
AUPR													
RUS		0.93 ± 0.21	0.90 ± 0.28	0.46 ± 0.25	0.29 ± 0.23	1.00 ± 0.00	0.70 ± 0.13	0.40 ± 0.08	0.20 ± 0.08	0.64 ± 0.20	0.34 ± 0.33	0.20 ± 0.23	0.10 ± 0.11
PRUS		**1.00** ± 0.00	**0.95** ± 0.14	**0.64** ± 0.30	**0.49** ± 0.35	1.00 ± 0.00	**0.77** ± 0.08	**0.43** ± 0.04	**0.26** ± 0.08	**0.88** ± 0.21	**0.81** ± 0.26	**0.62** ± 0.28	**0.37** ± 0.20
PCUS		**1.00** ± 0.00	**0.96** ± 0.05	**0.69** ± 0.24	**0.45** ± 0.29	1.00 ± 0.00	0.57 ± 0.19	0.38 ± 0.15	**0.27** ± 0.15	**0.99** ± 0.02	**0.86** ± 0.20	**0.76** ± 0.17	**0.53** ± 0.16
$PCUS_i$		0.90 ± 0.32	0.90 ± 0.32	**0.89** ± 0.31	**0.86** ± 0.31	0.90 ± 0.32	0.44 ± 0.24	0.35 ± 0.22	**0.29** ± 0.21	**1.00** ± 0.00	**0.97** ± 0.02	**0.96** ± 0.03	**0.80** ± 0.07

The boldface entries correspond to improvement in the performance of RUS after integrating progressive Boosting.

Table 7: Average of F_2-measure, G-mean and AUPR performance of proposed and baseline techniques on synthetic data over different levels of skew and overlap of test data.

Ensembles	Train Data	D_1 (λ_{train} =1:50, $\delta = 0.2$)				D_2 (λ_{train} =1:50, $\delta = 0.1$)				D_3 (λ_{train} =1:20, $\delta = 0.2$)			
	λ_{test}	**1:1**	**1:20**	**1:50**	**1:100**	**1:1**	**1:20**	**1:50**	**1:100**	**1:1**	**1:20**	**1:50**	**1:100**
F_2-measure													
Ada		0.97 ± 0.02	**0.97** ± 0.03	0.92 ± 0.02	0.78 ± 0.04	0.85 ± 0.05	0.75 ± 0.07	0.62 ± 0.05	0.42 ± 0.05	0.98 ± 0.02	0.93 ± 0.03	0.85 ± 0.02	0.57 ± 0.04
RUS		0.89 ± 0.11	0.85 ± 0.21	0.55 ± 0.19	0.34 ± 0.15	0.82 ± 0.09	0.62 ± 0.10	0.47 ± 0.08	0.27 ± 0.08	0.56 ± 0.20	0.35 ± 0.24	0.21 ± 0.18	0.13 ± 0.11
SMT		0.96 ± 0.04	0.96 ± 0.04	**0.94** ± 0.04	0.90 ± 0.03	0.94 ± 0.04	**0.85** ± 0.02	0.65 ± 0.02	0.61 ± 0.02	0.96 ± 0.04	0.91 ± 0.01	0.90 ± 0.02	0.74 ± 0.03
RB		0.96 ± 0.02	0.96 ± 0.02	**0.94** ± 0.02	0.90 ± 0.01	0.91 ± 0.05	**0.85** ± 0.01	0.63 ± 0.01	0.60 ± 0.01	0.96 ± 0.03	0.92 ± 0.02	0.90 ± 0.02	0.73 ± 0.05
CUS_i		0.95 ± 0.05	0.95 ± 0.05	**0.94** ± 0.04	0.89 ± 0.04	0.83 ± 0.02	0.71 ± 0.02	**0.67** ± 0.02	0.62 ± 0.02	**0.99** ± 0.02	**0.96** ± 0.02	0.87 ± 0.02	0.77 ± 0.01
PRUS-F		0.90 ± 0.07	0.76 ± 0.10	0.75 ± 0.09	0.74 ± 0.09	0.99 ± 0.02	0.54 ± 0.12	0.45 ± 0.08	0.44 ± 0.08	0.72 ± 0.06	0.63 ± 0.16	0.62 ± 0.16	0.57 ± 0.13
PCUS-F		0.80 ± 0.10	0.57 ± 0.22	0.54 ± 0.21	0.53 ± 0.21	0.91 ± 0.11	0.55 ± 0.19	0.46 ± 0.13	0.42 ± 0.12	0.87 ± 0.07	0.83 ± 0.08	0.78 ± 0.09	0.65 ± 0.13
$PCUS_i$-F		**0.98** ± 0.01	0.93 ± 0.03	0.92 ± 0.03	**0.91** ± 0.03	**1.00** ± 0.00	0.83 ± 0.03	**0.67** ± 0.03	**0.63** ± 0.04	0.93 ± 0.05	0.92 ± 0.05	**0.92** ± 0.05	**0.81** ± 0.05
G-mean													
Ada		0.97 ± 0.02	**0.97** ± 0.02	**0.97** ± 0.02	**0.97** ± 0.02	0.86 ± 0.05	0.86 ± 0.05	0.86 ± 0.05	0.85 ± 0.05	0.98 ± 0.02	**0.98** ± 0.02	**0.98** ± 0.02	0.97 ± 0.02
RUS		0.93 ± 0.05	0.93 ± 0.05	0.92 ± 0.05	0.91 ± 0.06	0.84 ± 0.08	0.83 ± 0.08	0.83 ± 0.08	0.82 ± 0.08	0.50 ± 0.20	0.68 ± 0.23	0.68 ± 0.23	0.68 ± 0.23
SMT		0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03	0.94 ± 0.04	**0.94** ± 0.04	**0.94** ± 0.04	**0.94** ± 0.04	0.96 ± 0.04	0.96 ± 0.03	0.96 ± 0.04	0.96 ± 0.04
RB		0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.02	0.96 ± 0.02	0.91 ± 0.05	0.91 ± 0.04	0.90 ± 0.04	0.91 ± 0.05	0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03
CUS_i		0.95 ± 0.04	0.95 ± 0.04	0.95 ± 0.04	0.95 ± 0.04	0.84 ± 0.01	0.84 ± 0.01	0.84 ± 0.01	0.84 ± 0.01	**0.99** ± 0.01	**0.98** ± 0.01	**0.98** ± 0.01	**0.98** ± 0.01
PRUS-F		0.90 ± 0.06	0.78 ± 0.08	0.78 ± 0.08	0.78 ± 0.08	0.99 ± 0.02	0.62 ± 0.09	0.62 ± 0.09	0.61 ± 0.09	0.78 ± 0.05	0.77 ± 0.05	0.76 ± 0.05	0.76 ± 0.06
PCUS-F		0.78 ± 0.20	0.65 ± 0.18	0.65 ± 0.18	0.65 ± 0.18	0.94 ± 0.07	0.68 ± 0.17	0.67 ± 0.17	0.64 ± 0.19	0.88 ± 0.06	0.86 ± 0.06	0.86 ± 0.06	0.86 ± 0.06
$PCUS_i$-F		**0.98** ± 0.01	0.93 ± 0.03	0.93 ± 0.03	0.93 ± 0.03	**1.00** ± 0.00	**0.94** ± 0.02	**0.94** ± 0.02	**0.94** ± 0.02	**0.94** ± 0.04	0.94 ± 0.04	0.94 ± 0.04	0.93 ± 0.05
AUPR													
Ada		**1.00** ± 0.00	**1.00** ± 0.00	0.98 ± 0.01	0.89 ± 0.04	**1.00** ± 0.00	0.82 ± 0.07	0.66 ± 0.05	0.34 ± 0.07	**1.00** ± 0.00	**0.99** ± 0.01	0.96 ± 0.01	0.70 ± 0.04
RUS		0.93 ± 0.21	0.90 ± 0.28	0.46 ± 0.25	0.29 ± 0.23	**1.00** ± 0.00	0.70 ± 0.13	0.40 ± 0.08	0.20 ± 0.08	0.64 ± 0.20	0.34 ± 0.33	0.20 ± 0.23	0.10 ± 0.11
SMT		0.90 ± 0.32	0.90 ± 0.32	0.89 ± 0.31	0.88 ± 0.31	0.90 ± 0.32	0.84 ± 0.30	0.52 ± 0.19	0.51 ± 0.19	**1.00** ± 0.00	**0.99** ± 0.01	**0.98** ± 0.01	0.88 ± 0.01
RB		0.80 ± 0.42	0.80 ± 0.42	0.79 ± 0.42	0.78 ± 0.41	0.60 ± 0.52	0.57 ± 0.49	0.32 ± 0.28	0.31 ± 0.27	**1.00** ± 0.00	**0.99** ± 0.01	**0.98** ± 0.01	0.84 ± 0.03
CUS_i		**1.00** ± 0.00	**1.00** ± 0.00	**1.00** ± 0.00	0.96 ± 0.01	0.99 ± 0.00	0.75 ± 0.04	**0.68** ± 0.01	**0.62** ± 0.02	**1.00** ± 0.00	**0.99** ± 0.01	0.96 ± 0.01	**0.92** ± 0.01
PRUS-F		**1.00** ± 0.00	**1.00** ± 0.00	0.99 ± 0.01	**0.97** ± 0.02	**1.00** ± 0.00	**0.86** ± 0.05	0.44 ± 0.04	0.35 ± 0.07	0.99 ± 0.01	**0.99** ± 0.01	0.90 ± 0.09	0.64 ± 0.15
PCUS-F		0.99 ± 0.04	0.92 ± 0.10	0.66 ± 0.27	0.47 ± 0.29	1.00 ± 0.00	0.57 ± 0.19	0.39 ± 0.15	0.27 ± 0.17	0.96 ± 0.07	0.87 ± 0.14	0.74 ± 0.17	0.53 ± 0.22
$PCUS_i$-F		**1.00** ± 0.00	**1.00** ± 0.00	0.98 ± 0.01	**0.97** ± 0.02	**1.00** ± 0.00	0.80 ± 0.09	0.55 ± 0.07	0.50 ± 0.09	**1.00** ± 0.00	0.98 ± 0.01	0.97 ± 0.01	0.85 ± 0.04

The boldface entries correspond to the best values of performance for each skew level of testing.

Table 8: Average of F_2-measure, G-mean and AUPR performance of baseline techniques before and after using the F-measure loss factor on FIA and COX data sets over different levels of skew in training and test data.

Ensembles	Train Data	FIA								COX							
		λ_{train} =1:50				λ_{train} =1:100				λ_{train} =1:50				λ_{train} =1:100			
	λ_{test}	1:1	1:20	1:50	1:100	1:1	1:20	1:50	1:100	1:1	1:20	1:50	1:100	1:1	1:20	1:50	1:100
F_2-measure																	
Ada		0.80 ± 0.18	0.80 ± 0.18	0.80 ± 0.18	0.79 ± 0.18	0.62 ± 0.29	0.62 ± 0.29	0.62 ± 0.29	0.62 ± 0.29	0.48 ± 0.30	0.48 ± 0.30	0.48 ± 0.30	0.48 ± 0.30	0.43 ± 0.25	0.43 ± 0.25	0.43 ± 0.25	0.43 ± 0.25
Ada-F		**0.99** ± 0.01	**0.99** ± 0.01	0.75 ± 0.03	0.75 ± 0.02	**0.87** ± 0.09	**0.74** ± 0.08	0.60 ± 0.05	0.62 ± 0.08	**0.99** ± 0.01	**0.99** ± 0.01	**0.55** ± 0.03	0.47 ± 0.02	**0.87** ± 0.09	**0.74** ± 0.08	**0.58** ± 0.05	0.42 ± 0.08
RUS		0.98 ± 0.02	0.86 ± 0.10	0.72 ± 0.15	0.57 ± 0.16	0.98 ± 0.02	0.85 ± 0.10	0.70 ± 0.16	0.59 ± 0.17	0.89 ± 0.11	0.50 ± 0.22	0.43 ± 0.22	0.37 ± 0.24	0.87 ± 0.12	0.44 ± 0.19	0.40 ± 0.19	0.36 ± 0.20
RUS-F		0.96 ± 0.04	**0.89** ± 0.03	0.72 ± 0.03	**0.58** ± 0.04	0.93 ± 0.04	0.85 ± 0.06	**0.71** ± 0.03	**0.60** ± 0.07	**0.96** ± 0.04	**0.89** ± 0.03	**0.55** ± 0.03	**0.42** ± 0.04	**0.93** ± 0.04	**0.71** ± 0.06	**0.51** ± 0.03	0.36 ± 0.07
SMT		0.93 ± 0.04	0.93 ± 0.04	0.93 ± 0.04	0.92 ± 0.05	0.92 ± 0.05	0.92 ± 0.05	0.92 ± 0.05	0.92 ± 0.05	0.81 ± 0.15	0.80 ± 0.15	0.80 ± 0.15	0.80 ± 0.15	0.82 ± 0.12	0.82 ± 0.12	0.82 ± 0.12	0.82 ± 0.12
SMT-F		**0.98** ± 0.02	**0.98** ± 0.02	0.92 ± 0.01	0.90 ± 0.02	**0.94** ± 0.04	0.90 ± 0.02	0.90 ± 0.02	0.90 ± 0.02	**0.98** ± 0.02	**0.98** ± 0.02	0.80 ± 0.01	0.80 ± 0.02	**0.94** ± 0.04	**0.85** ± 0.02	0.80 ± 0.02	0.80 ± 0.02
RB		0.89 ± 0.24	0.89 ± 0.24	0.89 ± 0.24	0.87 ± 0.24	0.77 ± 0.37	0.77 ± 0.37	0.76 ± 0.37	0.76 ± 0.37	0.83 ± 0.13	0.83 ± 0.13	0.83 ± 0.13	0.82 ± 0.13	0.82 ± 0.17	0.82 ± 0.17	0.82 ± 0.17	0.82 ± 0.17
RB-F		**0.99** ± 0.01	**0.99** ± 0.01	0.88 ± 0.01	0.85 ± 0.03	**0.91** ± 0.03	**0.85** ± 0.02	0.75 ± 0.02	0.74 ± 0.03	**0.99** ± 0.01	**0.99** ± 0.01	0.82 ± 0.01	0.81 ± 0.03	**0.91** ± 0.03	**0.85** ± 0.02	0.80 ± 0.02	0.79 ± 0.03
G-mean																	
Ada		0.82 ± 0.16	0.82 ± 0.16	0.82 ± 0.16	0.82 ± 0.16	0.67 ± 0.25	0.67 ± 0.25	0.67 ± 0.25	0.67 ± 0.25	0.56 ± 0.25	0.56 ± 0.25	0.56 ± 0.25	0.56 ± 0.25	0.52 ± 0.20	0.52 ± 0.20	0.52 ± 0.20	0.52 ± 0.20
Ada-F		**0.99** ± 0.01	**0.99** ± 0.01	**0.97** ± 0.01	**0.97** ± 0.01	**0.88** ± 0.08	**0.87** ± 0.07	**0.87** ± 0.07	**0.87** ± 0.07	**0.99** ± 0.01	**0.99** ± 0.01	**0.97** ± 0.01	**0.97** ± 0.01	**0.88** ± 0.08	**0.87** ± 0.07	**0.87** ± 0.07	**0.87** ± 0.07
RUS		0.97 ± 0.05	0.97 ± 0.02	0.97 ± 0.02	0.97 ± 0.02	0.95 ± 0.06	0.97 ± 0.02	0.97 ± 0.02	0.97 ± 0.02	0.85 ± 0.18	0.90 ± 0.07	0.94 ± 0.05	0.94 ± 0.05	0.83 ± 0.19	0.89 ± 0.08	0.93 ± 0.07	0.94 ± 0.07
RUS-F		0.96 ± 0.04	0.96 ± 0.04	0.95 ± 0.03	0.95 ± 0.03	0.93 ± 0.04	0.92 ± 0.04	0.92 ± 0.04	0.92 ± 0.04	**0.96** ± 0.04	**0.96** ± 0.04	**0.95** ± 0.03	**0.95** ± 0.03	**0.93** ± 0.04	**0.92** ± 0.04	0.92 ± 0.04	0.92 ± 0.04
SMT		0.94 ± 0.04	0.94 ± 0.04	0.94 ± 0.04	0.94 ± 0.04	0.92 ± 0.04	0.92 ± 0.04	0.92 ± 0.04	0.92 ± 0.04	0.83 ± 0.12	0.83 ± 0.12	0.83 ± 0.12	0.83 ± 0.12	0.84 ± 0.10	0.84 ± 0.10	0.84 ± 0.10	0.84 ± 0.10
SMT-F		**0.98** ± 0.02	**0.98** ± 0.02	**0.97** ± 0.02	**0.97** ± 0.02	**0.94** ± 0.03	**0.94** ± 0.03	**0.93** ± 0.03	**0.94** ± 0.03	**0.98** ± 0.02	**0.98** ± 0.02	**0.97** ± 0.02	**0.97** ± 0.02	**0.94** ± 0.03	**0.94** ± 0.03	**0.93** ± 0.03	**0.94** ± 0.03
RB		0.89 ± 0.24	0.89 ± 0.24	0.89 ± 0.24	0.89 ± 0.24	0.77 ± 0.37	0.77 ± 0.37	0.77 ± 0.37	0.77 ± 0.37	0.85 ± 0.11	0.85 ± 0.11	0.85 ± 0.11	0.85 ± 0.11	0.84 ± 0.16	0.84 ± 0.16	0.84 ± 0.16	0.84 ± 0.16
RB-F		**0.99** ± 0.01	**0.99** ± 0.01	**0.98** ± 0.01	**0.98** ± 0.01	**0.91** ± 0.03	**0.91** ± 0.03	**0.90** ± 0.03	**0.91** ± 0.03	**0.99** ± 0.01	**0.99** ± 0.01	**0.98** ± 0.01	**0.98** ± 0.01	**0.91** ± 0.03	**0.91** ± 0.03	**0.90** ± 0.03	**0.91** ± 0.03
AUPR																	
Ada		0.85 ± 0.35	0.82 ± 0.34	0.80 ± 0.33	0.77 ± 0.33	0.83 ± 0.36	0.79 ± 0.35	0.76 ± 0.35	0.74 ± 0.34	0.80 ± 0.36	0.66 ± 0.34	0.64 ± 0.34	0.60 ± 0.35	0.83 ± 0.32	0.65 ± 0.32	0.63 ± 0.33	0.61 ± 0.33
Ada-F		**1.00** ± 0.00	**1.00** ± 0.00	0.80 ± 0.06	0.77 ± 0.05	**1.00** ± 0.00	**0.81** ± 0.08	0.75 ± 0.13	0.72 ± 0.14	**1.00** ± 0.00	**1.00** ± 0.00	0.62 ± 0.06	0.60 ± 0.05	**1.00** ± 0.00	**0.81** ± 0.08	0.62 ± 0.13	0.60 ± 0.14
RUS		0.91 ± 0.29	0.88 ± 0.29	0.86 ± 0.28	0.85 ± 0.28	0.90 ± 0.30	0.88 ± 0.30	0.86 ± 0.29	0.85 ± 0.29	0.89 ± 0.28	0.81 ± 0.27	0.80 ± 0.27	0.79 ± 0.27	0.66 ± 0.46	0.57 ± 0.41	0.56 ± 0.41	0.56 ± 0.41
RUS-F		**1.00** ± 0.00	**0.96** ± 0.02	0.85 ± 0.04	0.84 ± 0.04	**1.00** ± 0.00	0.88 ± 0.11	0.86 ± 0.07	0.84 ± 0.09	**1.00** ± 0.00	**0.96** ± 0.02	0.79 ± 0.04	0.77 ± 0.04	**1.00** ± 0.00	**0.75** ± 0.11	0.57 ± 0.07	0.56 ± 0.09
SMT		1.00 ± 0.00	0.99 ± 0.01	0.99 ± 0.01	0.98 ± 0.02	1.00 ± 0.00	0.99 ± 0.01	0.99 ± 0.01	0.98 ± 0.02	0.99 ± 0.02	0.95 ± 0.05	0.95 ± 0.05	0.91 ± 0.07	0.99 ± 0.02	0.97 ± 0.05	0.96 ± 0.05	0.96 ± 0.05
SMT-F		**1.00** ± 0.00	**1.00** ± 0.00	0.96 ± 0.02	0.99 ± 0.01	**1.00** ± 0.00	0.93 ± 0.03	0.98 ± 0.04	0.97 ± 0.05	**1.00** ± 0.00	**1.00** ± 0.00	0.94 ± 0.02	0.90 ± 0.01	**1.00** ± 0.00	0.96 ± 0.03	0.95 ± 0.04	0.95 ± 0.05
RB		0.93 ± 0.26	0.92 ± 0.25	0.92 ± 0.25	0.91 ± 0.25	0.81 ± 0.39	0.80 ± 0.39	0.80 ± 0.39	0.79 ± 0.39	0.99 ± 0.03	0.94 ± 0.06	0.94 ± 0.06	0.91 ± 0.07	0.96 ± 0.17	0.93 ± 0.17	0.92 ± 0.17	0.92 ± 0.17
RB-F		**1.00** ± 0.00	**1.00** ± 0.00	0.91 ± 0.03	0.90 ± 0.02	**1.00** ± 0.00	**0.94** ± 0.02	0.79 ± 0.03	0.79 ± 0.03	**1.00** ± 0.00	**1.00** ± 0.00	0.93 ± 0.03	0.90 ± 0.02	**1.00** ± 0.00	**0.94** ± 0.02	0.91 ± 0.03	0.90 ± 0.03

The boldface entries correspond to improvement in the performance of the Boosting ensemble after modifying loss factor using F-measure.

Table 9: Average of F_2-measure, G-mean and AUPR performance of RUSBoost with and without integrating progressive Boosting FIA and COX data sets over different levels of skew in training and test data.

Ensembles	Train Data	FIA								COX							
		λ_{train} =1:50				λ_{train} =1:100				λ_{train} =1:50				λ_{train} =1:100			
	λ_{test}	1:1	1:20	1:50	1:100	1:1	1:20	1:50	1:100	1:1	1:20	1:50	1:100	1:1	1:20	1:50	1:100
F_2-measure																	
RUS		0.98 ± 0.02	0.86 ± 0.10	0.72 ± 0.15	0.57 ± 0.16	0.98 ± 0.02	0.85 ± 0.10	0.70 ± 0.16	0.59 ± 0.17	0.89 ± 0.11	0.50 ± 0.22	0.43 ± 0.22	0.37 ± 0.24	0.87 ± 0.12	0.44 ± 0.19	0.40 ± 0.19	0.36 ± 0.20
PRUS		**0.99** ± 0.01	**0.95** ± 0.04	**0.91** ± 0.06	**0.89** ± 0.06	**0.99** ± 0.01	**0.94** ± 0.04	**0.90** ± 0.06	**0.89** ± 0.07	**0.95** ± 0.06	**0.88** ± 0.10	**0.86** ± 0.09	**0.85** ± 0.10	**0.95** ± 0.06	**0.86** ± 0.13	**0.85** ± 0.13	**0.84** ± 0.13
PCUS		**0.99** ± 0.01	**0.95** ± 0.04	**0.91** ± 0.06	**0.88** ± 0.06	**0.99** ± 0.01	**0.95** ± 0.04	**0.91** ± 0.06	**0.89** ± 0.06	**0.95** ± 0.07	**0.88** ± 0.11	**0.85** ± 0.11	**0.84** ± 0.11	**0.95** ± 0.07	**0.85** ± 0.13	**0.84** ± 0.14	**0.84** ± 0.14
PTUS		**0.99** ± 0.01	**0.95** ± 0.04	**0.92** ± 0.05	**0.90** ± 0.06	**0.99** ± 0.01	**0.95** ± 0.04	**0.92** ± 0.06	**0.90** ± 0.06	**0.95** ± 0.06	**0.88** ± 0.10	**0.86** ± 0.10	**0.85** ± 0.11	**0.94** ± 0.07	**0.87** ± 0.12	**0.86** ± 0.12	**0.85** ± 0.13
G-mean																	
RUS		0.97 ± 0.05	0.97 ± 0.02	0.97 ± 0.02	0.97 ± 0.02	0.95 ± 0.06	0.97 ± 0.02	0.97 ± 0.02	0.97 ± 0.02	0.85 ± 0.18	0.90 ± 0.07	0.94 ± 0.05	0.94 ± 0.05	0.83 ± 0.19	0.89 ± 0.08	0.93 ± 0.07	0.94 ± 0.07
PRUS		**0.98** ± 0.03	**0.97** ± 0.03	**0.95** ± 0.04	0.94 ± 0.04	**0.97** ± 0.04	**0.97** ± 0.03	0.95 ± 0.03	0.94 ± 0.04	**0.95** ± 0.06	**0.92** ± 0.07	0.92 ± 0.07	0.91 ± 0.07	**0.95** ± 0.06	**0.91** ± 0.08	0.90 ± 0.09	0.90 ± 0.09
PCUS		**0.98** ± 0.03	**0.97** ± 0.02	**0.95** ± 0.03	0.94 ± 0.04	**0.97** ± 0.04	**0.97** ± 0.02	0.95 ± 0.03	0.94 ± 0.04	**0.95** ± 0.07	**0.91** ± 0.08	0.90 ± 0.08	0.90 ± 0.08	**0.95** ± 0.08	**0.91** ± 0.09	0.90 ± 0.09	0.89 ± 0.10
PTUS		**0.98** ± 0.03	**0.97** ± 0.03	**0.95** ± 0.04	0.94 ± 0.04	**0.98** ± 0.03	**0.97** ± 0.03	0.95 ± 0.04	0.94 ± 0.04	**0.95** ± 0.06	**0.91** ± 0.09	0.90 ± 0.09	0.88 ± 0.09	**0.94** ± 0.06	**0.91** ± 0.08	0.90 ± 0.09	0.89 ± 0.10
AUPR																	
RUS		0.91 ± 0.29	0.88 ± 0.29	0.86 ± 0.28	0.85 ± 0.28	0.90 ± 0.30	0.88 ± 0.30	0.86 ± 0.29	0.85 ± 0.29	0.89 ± 0.28	0.81 ± 0.27	0.80 ± 0.27	0.79 ± 0.27	0.66 ± 0.46	0.57 ± 0.41	0.56 ± 0.41	0.56 ± 0.41
PRUS		**1.00** ± 0.00	**0.95** ± 0.03	**0.93** ± 0.05	**0.91** ± 0.05	**1.00** ± 0.00	**0.95** ± 0.03	**0.93** ± 0.05	**0.91** ± 0.05	**0.98** ± 0.04	**0.88** ± 0.11	**0.85** ± 0.11	**0.83** ± 0.11	**0.83** ± 0.21	**0.72** ± 0.23	**0.71** ± 0.23	**0.71** ± 0.23
PCUS		**1.00** ± 0.00	**0.97** ± 0.03	**0.95** ± 0.04	**0.93** ± 0.05	**0.98** ± 0.14	**0.95** ± 0.14	**0.93** ± 0.14	**0.91** ± 0.14	**0.97** ± 0.05	**0.88** ± 0.12	**0.87** ± 0.12	**0.85** ± 0.13	**0.83** ± 0.21	**0.72** ± 0.23	**0.70** ± 0.23	**0.70** ± 0.24
PTUS		**0.98** ± 0.14	**0.96** ± 0.14	**0.94** ± 0.14	**0.92** ± 0.14	**0.97** ± 0.16	**0.95** ± 0.16	**0.93** ± 0.16	**0.91** ± 0.16	**0.97** ± 0.16	**0.88** ± 0.17	**0.86** ± 0.18	**0.84** ± 0.18	**0.84** ± 0.35	**0.76** ± 0.33	**0.75** ± 0.33	**0.73** ± 0.33

The boldface entries correspond to improvement in the performance of RUS after integrating progressive Boosting.

Table 10: Average of F_2-measure, G-mean and AUPR performance of proposed and baseline techniques FIA and COX data sets over different levels of skew in training and test data.

Ensembles	Train Data	FIA								COX							
		λ_{train} =1:50				λ_{train} =1:100				λ_{train} =1:50				λ_{train} =1:100			
	λ_{test}	1:1	1:20	1:50	1:100	1:1	1:20	1:50	1:100	1:1	1:20	1:50	1:100	1:1	1:20	1:50	1:100
F_2-measure																	
Ada		0.80 ± 0.18	0.80 ± 0.18	0.80 ± 0.18	0.79 ± 0.18	0.62 ± 0.29	0.62 ± 0.29	0.62 ± 0.29	0.62 ± 0.29	0.48 ± 0.30	0.48 ± 0.30	0.48 ± 0.30	0.48 ± 0.30	0.43 ± 0.25	0.43 ± 0.25	0.43 ± 0.25	0.43 ± 0.25
RUS		0.98 ± 0.02	0.86 ± 0.10	0.72 ± 0.15	0.57 ± 0.16	0.98 ± 0.02	0.85 ± 0.10	0.70 ± 0.16	0.59 ± 0.17	0.89 ± 0.11	0.50 ± 0.22	0.43 ± 0.22	0.37 ± 0.24	0.87 ± 0.12	0.44 ± 0.19	0.40 ± 0.19	0.36 ± 0.20
SMT		0.93 ± 0.04	**0.93** ± 0.04	**0.93** ± 0.04	**0.92** ± 0.05	0.92 ± 0.05	0.92 ± 0.05	**0.92** ± 0.05	**0.92** ± 0.05	0.81 ± 0.15	0.80 ± 0.15	0.80 ± 0.15	0.80 ± 0.15	0.82 ± 0.12	0.82 ± 0.12	0.82 ± 0.12	0.82 ± 0.12
RB		0.89 ± 0.24	0.89 ± 0.24	0.89 ± 0.24	0.87 ± 0.24	0.77 ± 0.37	0.77 ± 0.37	0.76 ± 0.37	0.76 ± 0.37	0.83 ± 0.13	0.83 ± 0.13	0.83 ± 0.13	0.82 ± 0.13	0.82 ± 0.17	0.82 ± 0.17	0.82 ± 0.17	0.82 ± 0.17
TUS		0.63 ± 0.18	0.63 ± 0.17	0.62 ± 0.17	0.62 ± 0.17	0.87 ± 0.08	0.87 ± 0.08	0.87 ± 0.08	0.86 ± 0.08	0.58 ± 0.27	0.49 ± 0.23	0.49 ± 0.23	0.48 ± 0.23	0.59 ± 0.27	0.59 ± 0.27	0.59 ± 0.27	0.59 ± 0.27
PRUS-F		**0.99** ± 0.01	0.95 ± 0.05	0.92 ± 0.06	0.90 ± 0.06	**0.99** ± 0.01	**0.95** ± 0.04	0.91 ± 0.06	0.90 ± 0.06	0.94 ± 0.07	0.87 ± 0.13	0.86 ± 0.14	0.85 ± 0.14	**0.95** ± 0.06	0.87 ± 0.12	0.86 ± 0.13	0.85 ± 0.14
PCUS-F		**0.99** ± 0.01	0.95 ± 0.04	0.92 ± 0.06	0.90 ± 0.06	**0.99** ± 0.02	**0.95** ± 0.04	0.91 ± 0.06	0.90 ± 0.07	**0.95** ± 0.07	0.85 ± 0.16	0.84 ± 0.17	0.82 ± 0.18	**0.95** ± 0.07	0.85 ± 0.16	0.84 ± 0.17	0.83 ± 0.17
PTUS-F		**0.99** ± 0.01	0.95 ± 0.04	0.92 ± 0.05	0.90 ± 0.06	**0.99** ± 0.01	**0.95** ± 0.04	**0.92** ± 0.06	0.90 ± 0.06	**0.95** ± 0.06	**0.89** ± 0.11	**0.87** ± 0.11	**0.86** ± 0.11	0.94 ± 0.06	**0.88** ± 0.11	**0.87** ± 0.11	**0.86** ± 0.11
G-mean																	
Ada		0.82 ± 0.16	0.82 ± 0.16	0.82 ± 0.16	0.82 ± 0.16	0.67 ± 0.25	0.67 ± 0.25	0.67 ± 0.25	0.67 ± 0.25	0.56 ± 0.25	0.56 ± 0.25	0.56 ± 0.25	0.56 ± 0.25	0.52 ± 0.20	0.52 ± 0.20	0.52 ± 0.20	0.52 ± 0.20
RUS		0.97 ± 0.05	**0.97** ± 0.02	**0.97** ± 0.02	**0.97** ± 0.02	0.95 ± 0.06	**0.97** ± 0.02	**0.97** ± 0.02	**0.97** ± 0.02	0.85 ± 0.18	0.90 ± 0.07	**0.94** ± 0.05	**0.94** ± 0.05	0.83 ± 0.19	0.89 ± 0.08	**0.93** ± 0.07	**0.94** ± 0.07
SMT		0.94 ± 0.04	0.94 ± 0.04	0.94 ± 0.04	0.94 ± 0.04	0.92 ± 0.04	0.92 ± 0.04	0.92 ± 0.04	0.92 ± 0.04	0.83 ± 0.12	0.83 ± 0.12	0.83 ± 0.12	0.83 ± 0.12	0.84 ± 0.10	0.84 ± 0.10	0.84 ± 0.10	0.84 ± 0.10
RB		0.89 ± 0.24	0.89 ± 0.24	0.89 ± 0.24	0.89 ± 0.24	0.77 ± 0.37	0.77 ± 0.37	0.77 ± 0.37	0.77 ± 0.37	0.85 ± 0.11	0.85 ± 0.11	0.85 ± 0.11	0.85 ± 0.11	0.84 ± 0.16	0.84 ± 0.16	0.84 ± 0.16	0.84 ± 0.16
TUS		0.49 ± 0.15	0.75 ± 0.13	0.76 ± 0.14	0.76 ± 0.14	0.88 ± 0.07	0.88 ± 0.07	0.88 ± 0.07	0.88 ± 0.07	0.70 ± 0.21	0.68 ± 0.20	0.69 ± 0.21	0.69 ± 0.21	0.71 ± 0.21	0.71 ± 0.21	0.71 ± 0.21	0.71 ± 0.21
PRUS-F		**0.98** ± 0.03	**0.97** ± 0.03	0.95 ± 0.04	0.94 ± 0.04	**0.98** ± 0.04	**0.97** ± 0.03	0.95 ± 0.03	0.94 ± 0.04	0.94 ± 0.08	**0.91** ± 0.08	0.91 ± 0.08	0.90 ± 0.09	**0.95** ± 0.07	**0.92** ± 0.07	0.91 ± 0.08	0.90 ± 0.09
PCUS-F		**0.98** ± 0.03	**0.97** ± 0.02	0.95 ± 0.04	0.94 ± 0.04	0.97 ± 0.04	**0.97** ± 0.02	0.95 ± 0.03	0.94 ± 0.04	**0.95** ± 0.07	**0.91** ± 0.08	0.90 ± 0.08	0.89 ± 0.08	**0.95** ± 0.08	0.91 ± 0.09	0.91 ± 0.09	0.90 ± 0.09
PTUS-F		**0.98** ± 0.02	**0.97** ± 0.03	0.95 ± 0.04	0.94 ± 0.04	**0.98** ± 0.03	**0.97** ± 0.02	0.95 ± 0.04	0.94 ± 0.04	**0.95** ± 0.08	**0.91** ± 0.09	0.90 ± 0.09	0.89 ± 0.09	0.94 ± 0.07	0.91 ± 0.08	0.90 ± 0.08	0.90 ± 0.09
AUPR																	
Ada		0.85 ± 0.35	0.82 ± 0.34	0.80 ± 0.33	0.77 ± 0.33	0.83 ± 0.36	0.79 ± 0.35	0.76 ± 0.35	0.74 ± 0.34	0.80 ± 0.36	0.66 ± 0.34	0.64 ± 0.34	0.60 ± 0.35	0.83 ± 0.32	0.65 ± 0.32	0.63 ± 0.33	0.61 ± 0.33
RUS		0.91 ± 0.29	0.88 ± 0.29	0.86 ± 0.28	0.85 ± 0.28	0.90 ± 0.30	0.88 ± 0.30	0.86 ± 0.29	0.85 ± 0.29	0.89 ± 0.28	0.81 ± 0.27	0.80 ± 0.27	0.79 ± 0.27	0.66 ± 0.46	0.57 ± 0.41	0.56 ± 0.41	0.56 ± 0.41
SMT		**1.00** ± 0.00	**0.98** ± 0.01	**0.96** ± 0.01	**0.94** ± 0.02	**1.00** ± 0.00	**0.99** ± 0.01	**0.97** ± 0.01	0.94 ± 0.02	**0.99** ± 0.02	**0.95** ± 0.05	**0.95** ± 0.05	**0.91** ± 0.07	**0.99** ± 0.02	**0.97** ± 0.05	**0.96** ± 0.05	**0.96** ± 0.05
RB		0.93 ± 0.26	0.92 ± 0.25	0.92 ± 0.25	0.91 ± 0.25	0.81 ± 0.39	0.80 ± 0.39	0.80 ± 0.39	0.79 ± 0.39	**0.99** ± 0.03	0.94 ± 0.06	0.94 ± 0.06	**0.91** ± 0.07	0.96 ± 0.17	0.93 ± 0.17	0.92 ± 0.17	0.92 ± 0.17
TUS		0.62 ± 0.15	0.62 ± 0.15	0.61 ± 0.15	0.60 ± 0.15	**1.00** ± 0.00	**0.99** ± 0.02	0.97 ± 0.03	**0.96** ± 0.03	**0.99** ± 0.02	0.40 ± 0.16	0.38 ± 0.17	0.37 ± 0.17	**0.99** ± 0.01	0.95 ± 0.05	0.93 ± 0.06	0.91 ± 0.08
PRUS-F		**1.00** ± 0.00	0.97 ± 0.04	0.95 ± 0.05	0.94 ± 0.06	**1.00** ± 0.00	0.98 ± 0.03	0.95 ± 0.04	0.94 ± 0.05	0.97 ± 0.06	0.88 ± 0.14	0.87 ± 0.14	0.86 ± 0.14	0.98 ± 0.04	0.88 ± 0.12	0.87 ± 0.14	0.86 ± 0.14
PCUS-F		**1.00** ± 0.00	0.98 ± 0.03	0.95 ± 0.05	0.93 ± 0.05	**1.00** ± 0.00	0.97 ± 0.03	0.95 ± 0.05	0.93 ± 0.06	0.97 ± 0.06	0.85 ± 0.18	0.83 ± 0.19	0.82 ± 0.20	0.97 ± 0.06	0.85 ± 0.18	0.83 ± 0.19	0.83 ± 0.20
PTUS-F		**1.00** ± 0.00	0.98 ± 0.03	0.96 ± 0.04	0.94 ± 0.05	**1.00** ± 0.00	0.98 ± 0.03	0.95 ± 0.04	0.94 ± 0.05	**0.99** ± 0.02	0.91 ± 0.09	0.89 ± 0.10	0.87 ± 0.11	**0.99** ± 0.02	0.91 ± 0.09	0.89 ± 0.11	0.88 ± 0.12

The boldface entries correspond to the best values of performance for each skew level of testing.

Table 11: Average ranking of the performance of proposed and baseline techniques.

Ensembles	Train Data	F-measure				G-mean				AUPR			
	λ_{test}	**1:1**	**1:20**	**1:50**	**1:100**	**1:1**	**1:20**	**1:50**	**1:100**	**1:1**	**1:20**	**1:50**	**1:100**
Ada		4.50	3.25	5.38	5.75	4.50	3.38	3.38	3.38	2.88	3.62	4.25	4.50
RUS		6.62	6.12	6.88	7.75	6.62	5.50	5.12	5.12	3.50	5.81	6.75	7.50
SMT		4.25	3.12	2.50	2.62	4.25	3.25	3.25	**3.12**	2.12	2.38	**2.12**	**2.00**
RB		4.38	3.38	3.88	4.12	4.50	3.38	3.38	3.50	5.56	6.31	5.62	5.00
TUS		5.12	4.38	4.00	3.88	5.12	4.25	4.25	4.12	2.31	4.06	3.38	3.00
PRUS-F		3.50	6.38	5.62	4.88	3.62	6.38	6.50	6.75	1.62	**2.00**	3.75	3.25
PCUS-F		5.88	6.50	6.12	5.38	5.62	6.88	7.00	6.62	3.00	5.94	6.88	6.88
PTUS-F		**1.75**	**2.88**	**1.62**	**1.62**	**1.75**	**3.00**	**3.12**	3.38	**1.50**	3.38	3.25	3.88

Table 12: P-values of statistical comparison between PTUS-F and baseline techniques.

Ensembles	Metric	F-measure				G-mean				AUPR			
	λ_{test}	**1:1**	**1:20**	**1:50**	**1:100**	**1:1**	**1:20**	**1:50**	**1:100**	**1:1**	**1:20**	**1:50**	**1:100**
Ada		**0.0125**	0.3821	**0.0011**	**0.0004**	0.1250	0.3821	0.4207	0.5000	0.1314	0.4207	0.2090	0.3050
RUS		**0.0010**	**0.0040**	**0.0000**	**0.0000**	**0.0000**	**0.0207**	0.0516	0.0778	0.0516	**0.0233**	**0.0022**	**0.0016**
SMT		**0.0207**	0.4207	0.2389	**0.0000**	**0.0207**	0.4207	0.4602	0.5000	0.3050	0.5000	0.5000	0.5000
RB		**0.0162**	0.3446	**0.0336**	0.2090	**0.0125**	0.3821	0.4207	**0.0398**	**0.0005**	**0.0084**	**0.0268**	0.1814
TUS		**0.0030**	0.3888	**0.0268**	**0.0336**	**0.0030**	0.1539	0.1814	0.2709	0.2546	0.2877	0.4602	0.5000