Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 460641, 15 pages
http://dx.doi.org/10.1155/2013/460641

Research Article

Optimizing Classification Decision Trees by Using Weighted Naïve Bayes Predictors to Reduce the Imbalanced Class Problem in Wireless Sensor Network

Hang Yang,[1] Simon Fong,[1] Raymond Wong,[2] and Guangmin Sun[3]

[1] *Department of Computer and Information Science, University of Macau, Taipa, Macau*
[2] *School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia*
[3] *Department of Electronic Engineering, Beijing University of Technology, Beijing 100022, China*

Correspondence should be addressed to Simon Fong; ccfong@umac.mo

Received 6 October 2012; Accepted 18 October 2012

Academic Editor: Sabah Mohammed

Standard classification algorithms are often inaccurate when used in a wireless sensor network (WSN), where the observed data occur in imbalanced classes. The imbalanced data classification problem occurs when the number of samples in one class, usually the class of interest, is much lower than the number in the other classes. Many classification models have been studied in the data-mining research community. However, they all assume that the input data are stationary and bounded in size, so that resampling techniques and postadjustment by measuring the classification cost can be applied. In this paper, we devise a new scheme that extends a popular stream classification algorithm to the analysis of WSNs for reducing the adverse effects of the imbalanced class in the data. This new scheme is resource light at the algorithm level and does not require any data preprocessing. It uses weighted naïve Bayes predictors at the decision tree leaves to effectively reduce the impact of imbalanced classes. Experiments show that our modified algorithm outperforms the original stream classification algorithm.

1. Introduction

A wireless sensor network (WSN) is a distributed platform that collects data over a broad area. It has a wide variety of practical military, medical, and industrial applications [1]. The brain of a WSN is usually a decision-making algorithm that is capable of correctly mapping a set of newly collected observations from the sensors to one or more predefined categories. It uses a machine-learning algorithm to recall the classification of old data and classify the new data accordingly. There is no shortage of machine-learning algorithms available for decision making in WSNs [2, 3]. However, the imbalanced classification is a common problem. This problem occurs when the classifier algorithm is trained with a dataset in which one class has only a few samples and there are a disproportionally large number of samples in the other classes. This kind of imbalanced data causes classifiers to be overfitted (i.e., produce redundant rules that describe duplicate or meaningless concepts) and as a result perform poorly, particularly in the identification of the minority class. In WSN applications, these rare minority classes are often critical. Some WSN examples include, but are not limited to, transaction fraud detection, machine fault monitoring, environmental anomalies, atypical medical conditions, and abnormal habitual behaviors—situations where the class of interest is a small sample of unusual readings. Studies [4] have shown that using standard classification algorithms to analyze these imbalanced class distributions leads to poor performance. An imbalanced class problem may have another implication in WSN where it could be a symptom of producing traffic “hot-spot” in WSN. The energy consumption in the sensors may become imbalanced too, which leads to premature drainout for some local nodes. Some solution [5] has been proposed to better cluster the nodes and traffics although it is aimed at the energy level.

Most of the standard classification algorithms assume that training examples are evenly distributed among different classes. In practical applications where this was known to be untrue, researchers addressed the problem by either manipulating the training data or adjusting the misclassification costs. Resizing the training datasets is a common strategy that attempts to downsize the majority class and oversamples the minority class. Many variants of this strategy have been proposed [6–8]. A second strategy is to adjust the costs of misclassification errors to be biased against or in favor of the majority and minority classes, respectively. Using the feedback from the altered error information, researchers then [9, 10] fine-tune their cost-sensitive classifiers and postprune the decision trees in the hope of establishing a balanced treatment of each class in the new imbalanced data collected by the network.

The authors of this paper argue that replacing the traditional classifier with an optimized stream classifier is another effective solution. As mentioned above, the current techniques for dealing with imbalanced data require additional data preprocessing or feedback learning and pruning of a trained decision tree. Though they may be useful in minimizing the impact of imbalanced data, these pre- and postprocessing mechanisms require working through a whole database and their operations incur certain overheads in the data-mining environment, which may not be favorable in a WSN. In a wireless sensor network, data mining is done in real time with a compact device with limited memory and processing power called a sink, and most importantly the incoming data for classification training and testing are streaming in nature. These data streams are nonstationary data that may only be read one time at the intermediate nodes of a sensor network and are then forgotten. Furthermore, these nodes may be required to perform real-time classification as the data flows along the WSN. Fast prediction results with satisfactory accuracy must be propagated from node to node. In this dynamic environment, techniques based on data storage and feedback style after learning cannot be used to correct imbalanced data. On the other hand, it has been demonstrated that a stream classifier is a good candidate for WSN applications [11].

The contribution of this paper is a set of simple modifications that optimize an existing stream classification algorithm called Very Fast Decision Tree to handle the imbalanced class problem at the algorithmic level. One important extension is the use of weighted naïve Bayes predictors installed at the decision tree leaves. The assigned weights have the effect of countering the "biases" that are introduced by the problems of imbalanced class found in imperfect WSN data. The paper is organized as follows. Section 2 describes in detail the modifications that tackle the imbalanced class problem. In Section 3 a range of experiments is described, the "biased" datasets with imbalanced classes are introduced and the experimental results are discussed. Section 4 concludes the paper.

2. Optimizing the Very Fast Decision Tree

2.1. Motivation and Overview. Three special modifications are proposed to enhance the Very Fast Decision Tree (VFDT) algorithm. These modifications are embedded in line with the codes that implement the classification logic of the stream classifier. The modifications to reduce the imbalanced class problem are made in four phases: the training phase, where new nodes are created if the statistical criteria established in the learning from the labeled samples phase are met; the prepruning phase, in which the qualified nodes and branches are tested to see whether they can indeed improve the prediction accuracy (before they are added to the decision tree); the prediction phase where unseen samples are being categorized to predefined classes; and the pruning phase which uses the functional tree leaf [12]. The modifications are in the forms of simple computation and conditional checks that do not incur heavy resource consumption at the sensor nodes.

The improved version of VFDT is generally called the Optimized Very Fast Decision Tree with Functional Leaves (OVFDT-FL). Our previous work [13] shows that the OVFDT-FL prototype can classify data streams with the maximum possible accuracy with the minimum tree size. In this paper, OVFDT-FL is tested with imbalanced class data. The design of OVFDT-FL is given as follows.

OVFDT, which is based on the original VFDT design, is implemented using a test-then-train approach for classifying a continuously arriving data stream, even $N \to \infty$ where N is the total number of training instances as shown in Figure 1. The whole test-then-train process is synchronized so that when the data stream arrives one segment at a time, the decision tree is tested first for prediction output and training (which is also known as updating) of the decision tree model then occurs incrementally. Suppose that X is a vector of d attributes and k is the number of classes included in the data streams. When a new data sample (X, y_k) arrives, it travels from the root of the decision tree to an existing leaf via the current decision tree structure, provided that the root existed initially. Otherwise, a heuristic function is used to construct a tree model with a single root node using the procedure shown in Pseudocode 1. Suppose that a decision tree model HT can give a prediction to a class $y_k^{'}$ according to the functional tree leaf $\mathscr{F}$, where $\mathrm{HT}(X) \to y_k^{'}$. Comparing the predicted class $y_k^{'}$ to the actual class y_k, the statistics of the true, C_T, and false, C_F, predictions are updated immediately. Meanwhile, the sufficient statistics n_{ijk}, which are a count of the attribute x_i with value j that belong to class y_k, are updated for each node. This series of actions is called the testing phase.

The training phase immediately follows the testing phase. Node-splitting estimation is used to initially decide if HT should be updated or not, depending on the number of samples received that can potentially be represented by additional underlying rules in the decision tree. In principle, the node-splitting estimation should apply to every single new sample that arrives. However, this would be too resource expensive and would slow down the tree building process. Instead, VFDT proposes a parameter $n_{\min}$ that only carries out the node-splitting estimation when $n_{\min}$ examples have been observed on a leaf. In the node-splitting estimation, the tree model should be updated when a heuristic function $G(\cdot)$ chooses the most appropriate attribute, with the highest heuristic function value

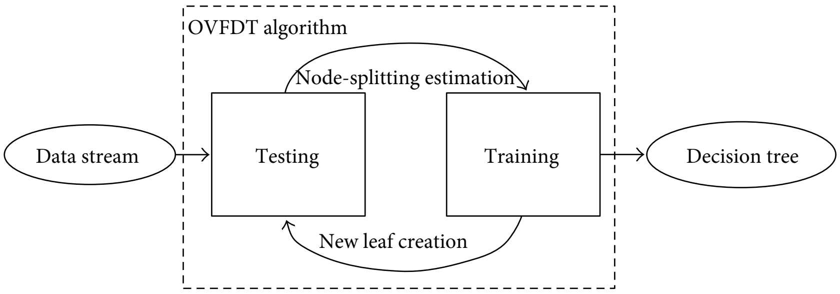


Figure 1: A test-then-train OVFDT workflow.

```
INPUT:
S: A stream of sample
X: A set of symbolic attributes
G(·): Heuristic function using for node-splitting estimation
δ: One minus the desired probability of choosing a correct attribute at any given node
$n_{\min}$: The minimum number of samples between check node-splitting estimation
$\mathscr{F}$: A functional tree leaf strategy
OUTPUT:
HT: A decision tree

PROCEDURE: initializeHT(S, X, G(·), δ, $n_{\min}$)
(1) Let HT be a tree with a single leaf l (the root). Let $X_l = X \cup \{X_\varnothing\}$
(2) Let $G_l\ (X_\varnothing)$ be the $G(\cdot)$ obtained by predicting the class in S, according to $\mathscr{F}$.
(3)  FOR each class $y_k$
(4) //$y_k$ is the class lable y with the kth label
(5)     FOR each value $X_{ij}$, of attribute $X_i \in X$
(6)            Reset OCD: $n_{ijk}(l) = 0$
(7)            //$n_{ijk}(l)$ is the count of attribute with $X_{ij}$ and $y_k$ at leaf l
(8)     END-FOR
(9)  END-FOR
(10) Return HT with a single root
```

Pseudocode 1: The pseudocode of initializing an OVFDT model.

$G(x_a)$, as a splitting node, according to Hoeffding's bound and the tie-breaking threshold. The heuristic function is implemented as an information gain here. This in situ system of node-splitting estimation constitutes our training phase.

Two modifications are proposed for the training phase of OVFDT to manage imbalanced data classes. The first is to dynamically adjust the tie-breaking function in the splitting-node determination using the mean value of Hoeffding's bound. The growth of the tree is influenced by the mean value of the traffic fluctuation (which was found to correlate with Hoeffding's bound in our previous work) rather than the imbalanced data class. The second modification is to use prepruning to test if the leaf chosen to be split, and therefore increase tree growth, is indeed a valid choice given the imbalanced data class. In this way, we can assume that the expansion of the tree is a result of genuinely accurate predictions. Thus, postpruning on the decision tree is not necessary. Section 2.2 presents the details of the functional leaf strategy for handling the imbalanced data class, and the details of the modifications to the training phase are given in Section 2.3.

2.2. Functional Tree Leaf Prediction in Testing Phase. The sufficient statistics n_{ijk} is an incremental count number stored in each node in the OVFDT. Suppose that a node Node_{ij} in HT is an internal node labeled with attribute x_{ij}. Suppose that k is the number of classes distributed in the training data, where $k \geq 2$. A vector V_{ij} is constructed from the sufficient statistics n_{ijk} in Node_{ij} such that $V_{ij} = \{n_{ij1}, n_{ij2}, \dots, n_{ijk}\}$. V_{ij} is the observed class distribution (OCD) vector of Node_{ij}. OCD stores the count of the distributed class at each tree node in OVFDT. It helps to keep track of the occurrences of the instances of each attribute.

For the actual classification, OVFDT uses $\text{HT}(X) \rightarrow y_k'$ to predict the class label when a new sample (X, y) arrives. The predictions are made according to the OCD in the leaves, which is called the functional tree leaf $\mathscr{F}$. Originally, in VFDT the prediction used only the majority class functional tree leaf

```
PROCEDURE: traverseHT(S, HT, $\mathscr{F}$)
(1) Sort S from the root to a leaf by HT. Update OCD in each node: $n_{ijk}(l)$ ++
(2) Switch ($\mathscr{F}$)
(3)     Case $\mathscr{F}^{\mathrm{MC}}$: predict the class $y_k'$ with max $n_{ijk}(l)$
(4)     Case $\mathscr{F}^{\mathrm{NB}}$: predict the class $y_k'$ with max NB prob.
(5)     Case $\mathscr{F}^{\mathrm{WNB}}$: predict the class $y_k'$ with max WNB prob.
(6)     Case $\mathscr{F}^{\mathrm{Adaptive}}$: predict the class $y_k'$ using $\mathscr{F}$ with $\mathrm{Error}_{\min}$
(7) IF $y_k'$ equals to the actual class label in S, THEN $C_T$++
(8) ELSE $C_F$++
(9) $\Delta C = C_T - C_F$
(10) Return $\Delta C$
```

Pseudocode 2: The pseudocode of OVFDT testing phase.

$\mathscr{F}^{\mathrm{MC}}$. The majority class only considers the counts of the class distribution, but not the decisions based on combinations of attributes. The naïve Bayes functional tree leaf $\mathscr{F}^{\mathrm{NB}}$ was proposed to compute the conditional probabilities of the attribute values given a class at the tree leaves by naïve Bayes. As a result, the prediction at the leaf is refined by the consideration of the probabilities of each attribute. To handle imbalanced class distribution in a stream, a weighted naïve Bayes functional tree leaf $\mathscr{F}^{\mathrm{WNB}}$ and an adaptive functional tree leaf $\mathscr{F}^{\mathrm{Adaptive}}$ are proposed in the paper.

2.2.1. Majority Class Functional Tree Leaf. In the ODC vector, the majority class functional tree Leaf $\mathscr{F}^{\mathrm{MC}}$ chooses the class with the maximum distribution as the predictive class in a leaf, where $\mathscr{F}^{\mathrm{MC}}$: $\arg\max f = \{n_{i,j,1}, n_{i,j,2}, \dots, n_{i,j,r}, \dots, n_{i,j,k}\}$, and where $0 < r < k$.

2.2.2. Naïve Bayes Functional Tree Leaf. In the OCD vector $V_{i,j} = \{n_{i,j,1}, n_{i,j,2}, \dots, n_{i,j,r}, \dots, n_{i,j,k}\}$, where r is the number of observed classes and $0 < r < k$, the naïve Bayes functional tree leaf $\mathscr{F}^{\mathrm{NB}}$ chooses the class with the maximum possibility, as computed by the naïve Bayes, as the predictive class in a leaf. $n_{i,j,r}$ is updated to $n_{i,j,r}'$ by the naïve Bayes function such that $n_{i,j,r}' = \mathrm{P}(X \mid C_f) \cdot \mathrm{P}(C_f)/\mathrm{P}(X)$, where X is the new arrival instance. Hence, the prediction class is $\mathscr{F}^{\mathrm{NB}}$: $\arg\max i = \{n_{i,j,1}', n_{i,j,2}', \dots, n_{i,j,r}', \dots, n_{i,j,k}'\}$.

2.2.3. Weighted Naïve Bayes Functional Tree Leaf. In the OCD vector $V_{i,j} = \{n_{i,j,1}, n_{i,j,2}, \dots, n_{i,j,r}, \dots, n_{i,j,k}\}$, where k is the number of observed classes and $0 < r < k$, the weighted naïve Bayes functional tree leaf $\mathscr{F}^{\mathrm{WNB}}$ chooses the class with the maximum possibility, as computed by the weighted naïve Bayes, as the predictive class in a leaf. $n_{i,j,r}$ is updated to $n_{i,j,r}'$ by the weighted naïve Bayes function such that $n_{i,j,r}' = \omega_r \cdot \mathrm{P}(X \mid C_f) \cdot \mathrm{P}(C_f)/\mathrm{P}(X)$, where X is the new arrival instance, and the weight is the probability of class i distribution amongst all the observed samples such that $\omega_r = \prod_{r=1}^{k}(\upsilon_r/\sum_{r=1}^{k}\upsilon_r)$, where $n_{i,j,r}$ is the count of class r. Hence, the prediction class is $\mathscr{F}^{\mathrm{WNB}}$: $\arg\max f = \{n_{i,j,1}', n_{i,j,2}', \dots, n_{i,j,r}', \dots, n_{i,j,k}'\}$.

2.2.4. Adaptive Functional Tree Leaf. In a leaf, suppose that $V_{\mathscr{F}^{\mathrm{MC}}}$ is the observed class distribution vector with the majority class functional tree leaf $\mathscr{F}^{\mathrm{MC}}$, suppose that $V_{\mathscr{F}^{\mathrm{NB}}}$ is the observed class distribution vector with the naïve Bayes functional tree leaf $\mathscr{F}^{\mathrm{NB}}$, and suppose that $V_{\mathscr{F}^{\mathrm{WVB}}}$ is the observed class distribution vector with the weighted naïve Bayes functional tree leaf $\mathscr{F}^{\mathrm{WNB}}$. Suppose that y is the true class of a new instance X. Suppose that $E_{\mathscr{F}}$ is the prediction error rate using a functional tree leaf $\mathscr{F}$. $E_{\mathscr{F}}$ is calculated by the average $E = \mathrm{error}_i/n$, where n is the number of examples and error_i is the number of examples mispredicted using $\mathscr{F}$. The adaptive functional tree leaf chooses the class with the minimum error rate predicted by the other three strategies, where $\mathscr{F}^{\mathrm{Adaptive}}$: $\arg\min \mathscr{F} = \{E_{\mathscr{F}^{\mathrm{MC}}}, E_{\mathscr{F}^{\mathrm{NB}}}, E_{\mathscr{F}^{\mathrm{WVB}}}\}$.

According to the functional tree leaf strategy, the current HT sorts a newly arrived sample (X, y_k) from the root to a predicted leaf y_k'. Comparing the predicted class y_k' to the actual class y_k, the statistics of truly C_T and falsely C_F prediction are updated immediately. C_T and C_F are used in the model-training phase. Pseudocode 2 is a flowchart of the modified testing phase.

2.3. Dynamic Splitting Test and Prepruning in the Training Phase. The node-splitting control is modified to use a dynamic tie-breaking threshold τ, which restricts the attribute splitting at a decision node. The τ parameter traditionally is preconfigured with a default value defined by the user. The optimal value is usually not known until all of the possibilities in an experiment have been tried. Longitudinal testing of different values in advance is certainly not favorable in real-time applications. Instead, we assign a dynamic tie threshold, equal to the dynamic mean of the HB value at each pass of stream data, as the splitting threshold, which controls the node splitting during the tree-building process. Tie breaking that occurs close to the HB mean can effectively narrow the variance distribution. The HB mean is calculated dynamically whenever new data arrives and the HB value is updated.

The estimation of splits and ties is only executed once for every $n_{\min}$ (a user-supplied value) sample that arrives at a leaf. Instead of a pre-configured tie, OVFDT uses an adaptive tie that is calculated by incremental computing. At the ith node-splitting estimation, Hoeffding's bound ε estimates whether there are sufficient statistics from a large enough sample size to split a new node, which corresponds to the leaf l. Let T_l be an adaptive tie corresponding to leaf l, within k estimations seen so far. Suppose that μ_l is a binary variable that takes the value of 1 if ε relates to leaf l and 0 otherwise. T_l is computed by (1). To constrain HB fluctuation, an upper bound T_l^{UPPER} and a lower bound T_l^{LOWER} are proposed in the adaptive tie mechanism. The formulas are presented in (2) and (3) as follows:

$$\mathrm{T}_l = \frac{1}{k}\sum_{i=1}^{k}\mu_l \times \varepsilon_i, \qquad (1)$$

$$T_l^{\mathrm{UPPER}} = \arg\max \mathrm{T}_l, \qquad (2)$$

$$T_l^{\mathrm{LOWER}} = \arg\min \mathrm{T}_l. \qquad (3)$$

For resource-light operations, we propose an error-based prepruning mechanism for the OVFDT, which stops noninformative node splitting before it splits into a new node. The prepruning takes into account both global and local node-splitting errors.

Lemma 1 (Monitoring Global Accuracy). *The model's accuracy varies whenever a node splits and the tree structure is updated. Overall accuracy of a current tree model is monitored during node splitting by comparing the number of correctly and incorrectly predicted samples. The numbers of correctly predicted instances and otherwise are recorded as current global performance indicators. This monitoring allows the determination of global accuracy.*

When a new instance arrives, it will be sorted to a leaf by the current HT structure before the node-splitting estimation. This is the "testing" phase in OVFDT. Suppose that C_T is the number of correctly predicted instances in the current HT and C_F is the number of incorrectly predicted instances. After the ith node-splitting estimation, let ΔC_i be the difference between C_T and C_F, then ΔC_i is computed by (4), which reflects the global accuracy of the current HT prediction on the newly arrived data streams. If $\Delta C_i \geq 0$, the number of correct predictions is no less than the number of incorrect predictions in the current tree structure; otherwise, the current tree graph needs to be updated by node splitting.

Lemma 2 (Monitor Local Accuracy). *The global accuracy can be tracked by comparing the number of correctly predicted samples with the number of incorrectly predicted samples. Likewise, comparing the global accuracy as measured at the current node-splitting estimation with the global accuracy measured at the previous splitting, means that the variation in accuracy is being tracked dynamically. This monitoring allows us to check whether the current node splitting is advantageous at each step by comparing it with the previous step.*

Suppose that $\mathrm{Gain}_{\mathrm{Accu}}$ is the gain in accuracy of the ith and the $(i-1)$th estimations, as calculated in (5), which reflects a local accuracy of changes. If $\mathrm{Gain}_{\mathrm{Accu}}(\mathrm{HT}_i) \geq 0$, the measurement of accuracy at the ith splitting HT structure is no worse than the accuracy at the $(i-1)$th splitting; otherwise, the old tree structure needs to be updated. The splitting estimation is implemented once for every $n_{\min}$ sample that arrives at a leaf. The tree size increases by l when a new node splits. The number of samples that meets the first pruning condition is $(n_{\min} \cdot p)$, where p is the probability of the optimal node splitting calculated in (8). Only one value of p can be chosen at one splitting estimation. The calculation of tree size at estimation i is given in (6). C_T and C_F in the ith splitting estimation give feedback on the tree's current classifying accuracy. By continually comparing this with $(i-1)$th, the pruning maintains the accuracy sequentially. In other words, the optimum result is obtained by comparing the current tree status to its previous status as follws:

$$\Delta C_i = C_T - C_F, \qquad (4)$$

$$\mathrm{Gain}_{\mathrm{Accu}}(\mathrm{HT}_i) = \Delta C_i - \Delta C_{i-1}, \qquad (5)$$

$$L_i = L_{i-1} + n_{\min} \cdot p, \qquad (6)$$

$$\mathrm{Gain}_{\mathrm{Tree\ Size}}(\mathrm{HT}_i) = L_i - L_{i-1}, \quad (L_0 = 1), \qquad (7)$$

$$p = \begin{cases} \mathrm{Prob}\left[\Delta G \leq T_l^{\mathrm{LOWER}}\right] \cdot \mathrm{Prob}\left[\Delta C_i < \Delta C_{i-1}\right] \ \text{ or} \\ \mathrm{Prob}\left[\Delta G \leq T_l^{\mathrm{LOWER}}\right] \cdot \mathrm{Prob}\left[\Delta C_i < 0\right] \ \text{ or} \\ \mathrm{Prob}\left[T_l^{\mathrm{LOWER}} < \Delta G < T_l^{\mathrm{UPPER}}\right] \cdot \mathrm{Prob}\left[\Delta C_i < \Delta C_{i-1}\right]. \end{cases} \qquad (8)$$

Figure 2 shows why our proposed prepruning takes into account both the local and the global accuracy in the incremental pruning. At the ith node-splitting estimation, the difference between correctly and incorrectly predicted classes was ΔC_i, and ΔC_{i+1} at the $i+1$th estimation. $\mathrm{Gain}_{\mathrm{Accu}}(\mathrm{HT}_{i+1})$ was negative, indicating that the local accuracy of $i+1$th estimation was worse than that at the previous node-splitting, although both were on a globally increasing trend. Thus, if accuracy is declining locally, it is necessary to update the HT structure even if accuracy is increasing globally.

The optimal node splitting control consists of a dynamic tie for node splitting and a prepruning mechanism that tries to hold the tree growth in neutral with respect to the imbalanced class distribution. In each node-splitting estimation process, the Hoeffding bound (HB) value that relates to leaf l is recorded. The recorded HB values are used to compute the adaptive tie, which uses the mean of the values for each leaf l instead of a fixed user-defined value as in VFDT. Using all the prediction statistics gathered in the testing phase for implementing prepruning, Pseudocode 3 presents the pseudocode of the training phase used by OVFDT for building an upright tree.

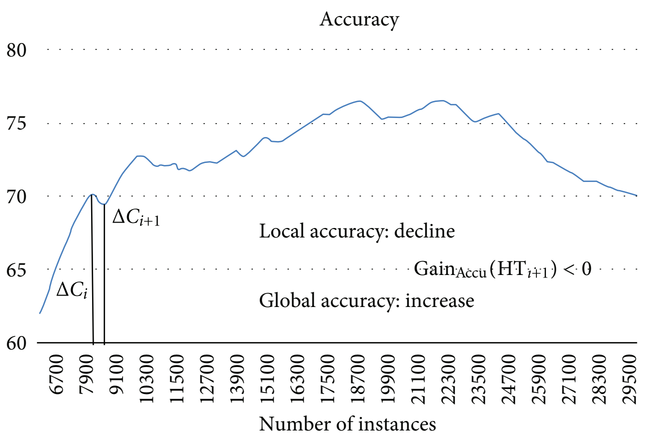


FIGURE 2: Example of incremental pruning.

3. Experiments

3.1. Bias Generator. For our experiments, we adopted and customized massive online analysis (MOA), one of the most popular data stream-mining toolkits, by including the aforementioned modifications into the OVFDT algorithm. However, the latest version of the MOA simulation environment is not able to simulate a biased data stream with an imbalanced class. A bias generator was therefore written in JAVA code and integrated into MOA for the purpose of evaluating the performance of stream-mining algorithms under imbalanced class data. Using either a simple command-line console or a graphic user, of which an example is shown in Figure 3, the generator injected biased instances from a specific imbalanced class into a given ARFF file. The input parameters are as follows:

(i) *biased class index* (BCI): the class index that the bias-added instances belong to,

(ii) *bias change from class index* (CCI): the class index that the bias instances will replace,

(iii) *change reduction percentage* (CP): the proportion of instances that will change to biased instances.

After the generator configuration, the instances with CCI class are replaced by BCI instances according to the CP setting. For example, in the snapshot below BCI = 5, CCI = 4, and CP = 80%; this means that 80% of CCI instances are replaced by BCI instances (the original settings are BCI = 10% and CCI = 10%, after BCI = 18% and CCI = 2%).

3.2. Experiment Datasets and Visualization. Six datasets were used to test the performance of OVFDT + FL versus ordinary VFDT. The datasets included those generated by the biased simulator and naturally imbalanced real-life data downloaded from the UCI machine-learning archive (http://www.ics.uci.edu/~mlearn). Table 1 describes these experimental datasets in detail, and Figure 4 provides a group of class distribution visualizations.

The following charts visualize the bias-included datasets with the imbalanced class. The pie charts on the left show the class distribution in the full experimental datasets and the charts on the right show the class distribution being progressively updated as new data streams arrive. For example, from Table 1 we see that the biased classes in the LED24 dataset are Class 2 (18%) and Class 4 (18%). There are 80% more data samples for these two classes than for the other classes. Originally the data distributions over all classes were equal. The charts representing the other datasets show at least one class, which has a larger percentage of data distribution than others.

3.3. Experiment Results Comparing VFDT and OVFDT. VFDT is deemed to be a suitable candidate for real-time classification in wireless sensor networks, because of its incremental learning nature based on a test-and-train approach. In this paper, we extend the design of VFDT to OVFDT, which has superior mechanisms for dealing with imbalanced data classes. This following comparison is between VFDT and OVFDT, which use the same types of functional tree leaf in the imbalanced datasets. The goal is to observe the comparative impact of the imbalanced classes on VFDT and OVFDT. For VFDT, the fixed tie breaking threshold (range from 0 to 1) is an important predefined parameter τ, which controls the node-splitting speed. In the experiment, τ was set at different values from 0.1 to 1.0 to test several different trails of VFDT, as a priori information for τ values is unavailable until the model is actually put to the test. The number of correctly classified instances measures the accuracy over the total number of arrived instances.

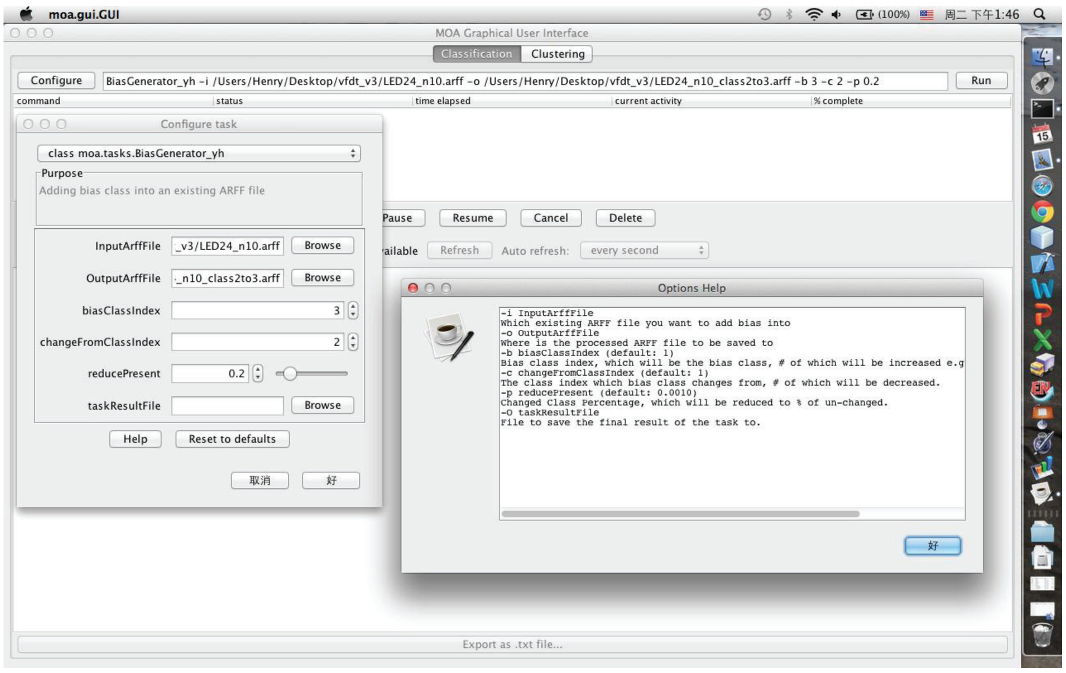
The results show, on the one hand, that $\text{OVFDT}_{\text{Adaptive}}$ has better performance results than any other method, for imbalanced data streams. The highlighted areas in Figure 5 show that OVFDT consistently outperformed VFDT. OVFDT_{MC} had lower accuracy than other functional tree leaf strategies in OVFDT. The advantage of the functional tree leaf approach is more apparent in the analysis of imbalanced data streams that have a significantly large bias in class distribution. This means that the modification at the testing phase is substantially effective, even when processing highly imbalanced data classes. On the other hand, the advantage of the other two modifications to the training phase, prepruning and dynamic node splitting, shows their usefulness in reducing the overfitting problem caused by imbalanced class data streams.

The radar chart in Figure 6 demonstrates that OVFDT results in a much smaller tree size than VFDT in all cases. A small tree size means lower runtime memory requirements, which makes it suitable for operating sensor node devices in WSNs. Tree size is measured by the number of leaves in a decision tree. Ideally there should be just enough leaves and corresponding branch paths to correctly classify the samples. Having too many leaves is a symptom of overfitting, which results in a decision tree that cannot make meaningful predictions and uses up memory space.

As these experimental results show, OVFDT with a functional tree leaf handles imbalanced data streams more effectively than VFDT. For this reason, VFDT will not be considered in the following experiments. Instead, we will

```
PROCEDURE doNodeSplittingEstimation($\Delta C$, $S$, $X$, $G(\cdot)$, $\delta$)
(1) FOR each attribute $X_i \in X_l - \{X_\varnothing\}$ at the leaf $l$
(2)     Compute $G_l(X_i)$
(3)     Let $X_a$ be the attribute with highest $G_l(\cdot)$ and $X_b$, with the 2nd highest $G_l(\cdot)$
(4)     Compute HB with $\delta$
(5)     Let $\Delta G_l = G_l(X_a) - G_l(X_b)$
(6) END-FOR
(7) IF ($\Delta G_l >$ HB) or ($\Delta G \le T_l^{\text{LOWER}}$ and $\Delta C_i < \Delta C_{i-1}$) or ($\Delta G \le T_l^{\text{LOWER}}$ and $\Delta C_i < 0$) or ($T_l^{\text{LOWER}} < \Delta G \le T_l^{\text{UPPER}}$ and $\Delta C_i < \Delta C_{i-1}$)
(8)     Replace $l$ by an internal node splits on $X_a$
(9)     Update Adaptive tie $T_l^{\text{LOWER}}$ and $T_l^{\text{UPPER}}$
(10)    FOR each branch of splitting
(11)        Add a new leaf $l_m$ and let $X_m = X - \{X_a\}$
(12)        Let $G(X_\varnothing)$ be $G(\cdot)$ obtained by predicting the class in $S$, according to $\mathcal{F}$ at $l_m$
(13)        FOR each class $y_k$ and each value $x_{ij}$ of each attribute
(14)            $X_i \in X_m - \{X_\varnothing\}$ and reset OCD: $n_{ijk}(l) = 0$
(15)        END-FOR
(16)    END-FOR
(17) END-IF
(18) Return updated HT
```

Pseudocode 3: The pseudocode of OVFDT model training.



```
BiasGenerator_yh -i /Users/Henry/Desktop/vfdt_v3/LED24_n10.arff -o /Users/Henry/Desktop/vfdt_v3/LED24_n10_class2to3.arff -b 3 -c 2 -p 0.2
```

class moa.tasks.BiasGenerator_yh

Purpose: Adding bias class into an existing ARFF file

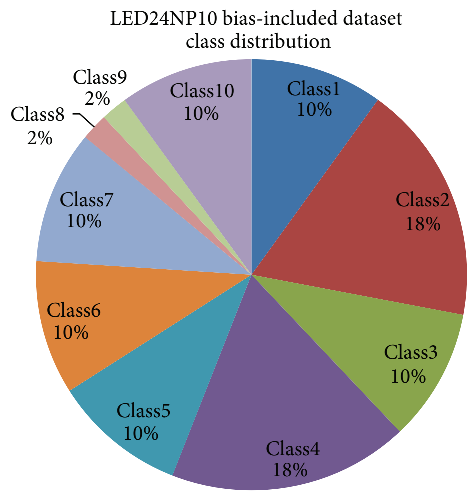
InputArffFile	_v3/LED24_n10.arff
OutputArffFile	·_n10_class2to3.arff
biasClassIndex	3
changeFromClassIndex	2
reducePresent	0.2
taskResultFile	

```
-i InputArffFile
Which existing ARFF file you want to add bias into
-o OutputArffFile
Where is the processed ARFF file to be saved to
-b biasClassIndex (default: 1)
Bias class index, which will be the bias class, # of which will be increased e.g
-c changeFromClassIndex (default: 1)
The class index which bias class changes from, # of which will be decreased.
-p reducePresent (default: 0.0010)
Changed Class Percentage, which will be reduced to % of un-changed.
-O taskResultFile
File to save the final result of the task to.
```

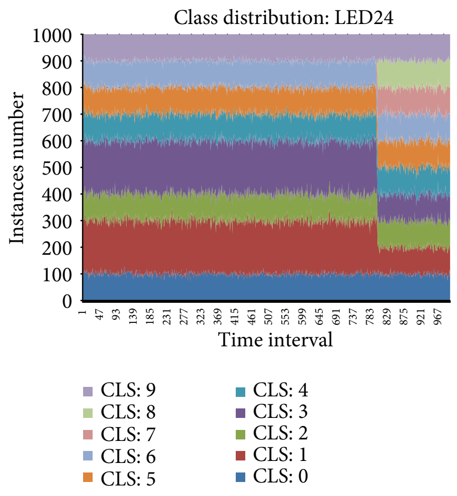
Figure 3: Snapshot of the Bias generator for generating data with imbalanced class, on MOA platform.

Table 1: Datasets with imbalanced class used in the experiment.

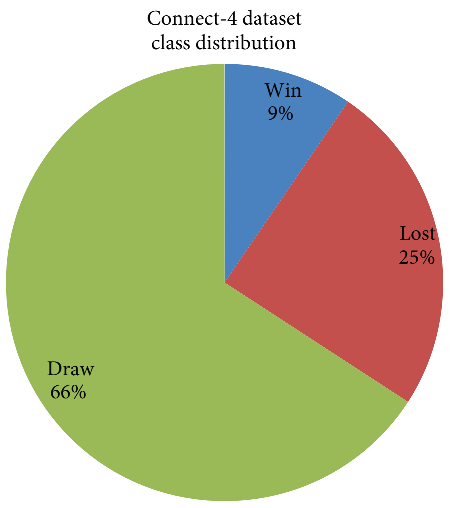
Name	Type	Source	Nom. attr. no.	Num. attr. no.	CLS no.	Bias CLS	Inst no.
LED24	Nominal	Synthetic	24	0	10	2, 4	10^6
Connect-4	Nominal	UCI	42	0	7	“Draw”	67,557
Waveform 21	Numeric	Synthetic	0	21	3	1	10^6
Radial bias function (RBF)	Numeric	Synthetic	0	50	10	1, 3, 5, 8, 10	10^6
Random tree (RT)	Mixed	Synthetic	50	50	10	4, 5, 10	10^6
COVTYPE	Mixed	UCI	42	12	7	2, 7	581,012



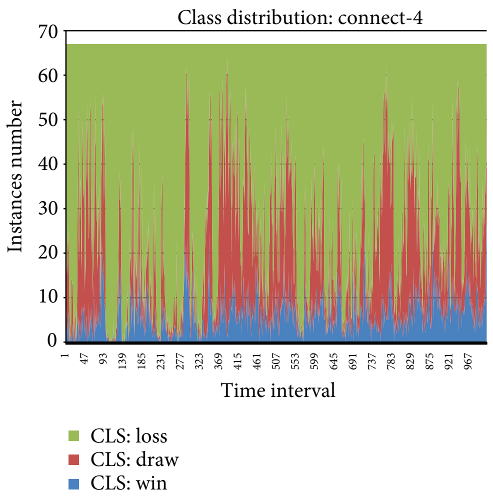
(a)



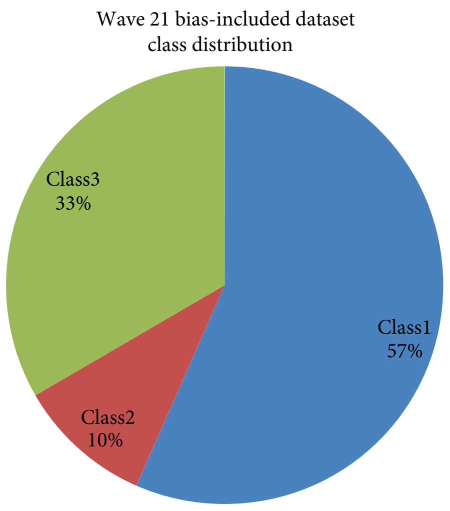
(b)



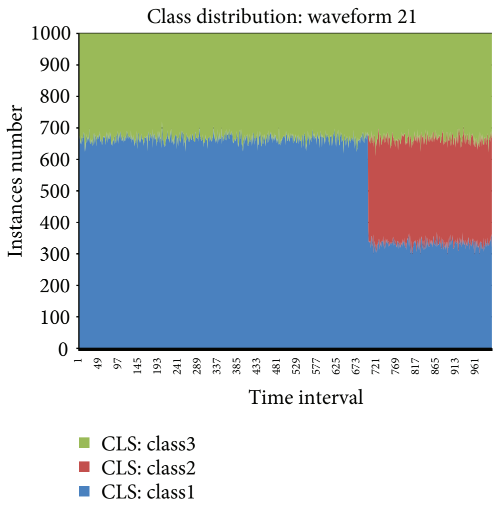
(c)



(d)

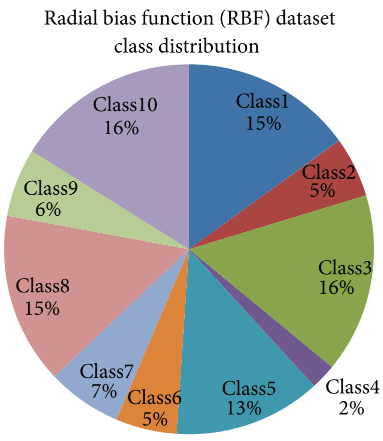


(e)

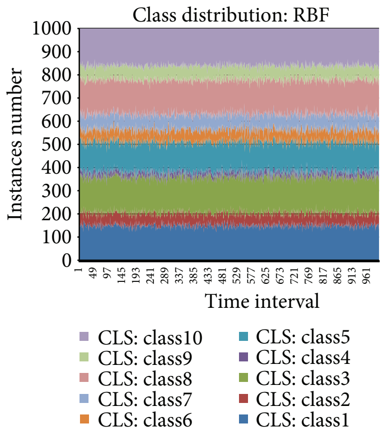


(f)

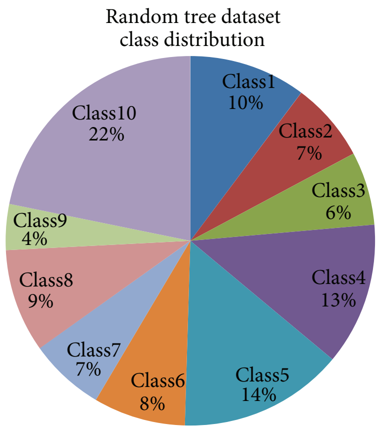
FIGURE 4: Continued.



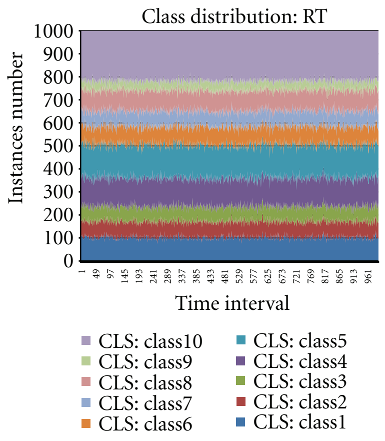
(g)



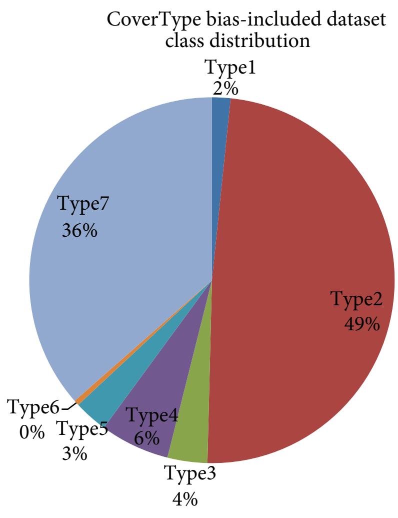
(h)



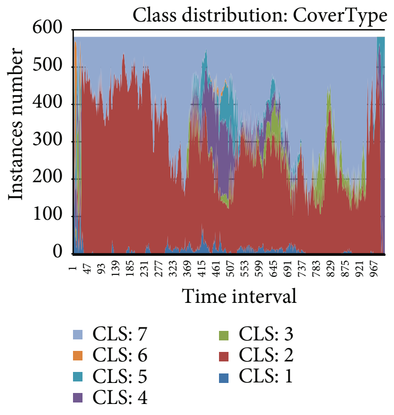
(i)



(j)



(k)



(l)

Figure 4: A collection of visualizations of the datasets that have different degrees of imbalanced class distribution.

analyze in detail the experimental results of OVFDT using different types of functional tree leaves.

3.4. Experiment Results Comparing OVFDT Functional Tree Leaf Accuracy. Comparing the classification accuracy of four different types of functional tree leaves, we find that OVFDT$_{MC}$ always obtains the lowest accuracy and OVFDT$_{Adaptive}$ has consistently better accuracy than the other methods. In addition, OVFDT$_{WNB}$ is better than OVFDT$_{NB}$ in experiments that weight the probabilities of each attribute occurrence (see Figure 8).

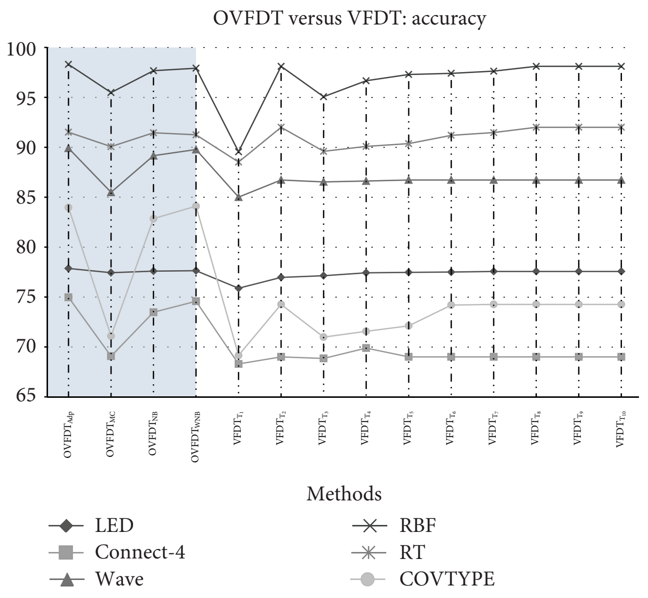


FIGURE 5: Accuracy of the classification experiments by VFDT and OVFDT with datasets of imbalanced data class.

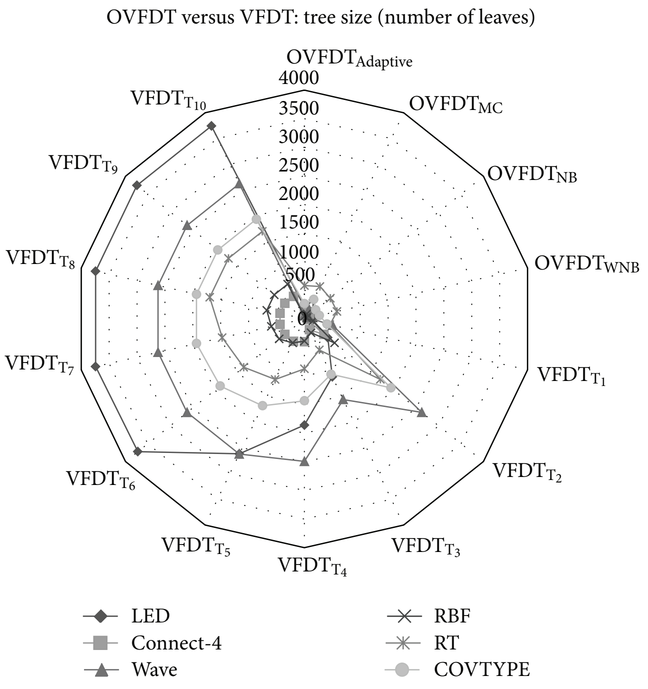


FIGURE 6: Tree size of the classification experiments by VFDT and OVFDT with datasets of imbalanced data class.

Prediction outcomes

Actual values	TP$_1$ C_{11}	FN$_1$ C_{12}	C_{13}	$\ldots$	C_{1d}
	FP$_1$ C_{21}	TN$_1$ C_{22}	C_{23}	$\ldots$	C_{2d}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	C_{d1}	C_{d2}	C_{d3}	$\ldots$	C_{dd}

FIGURE 7

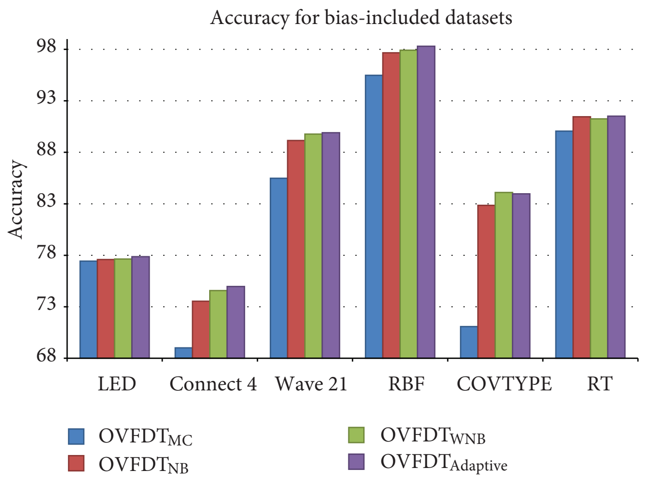


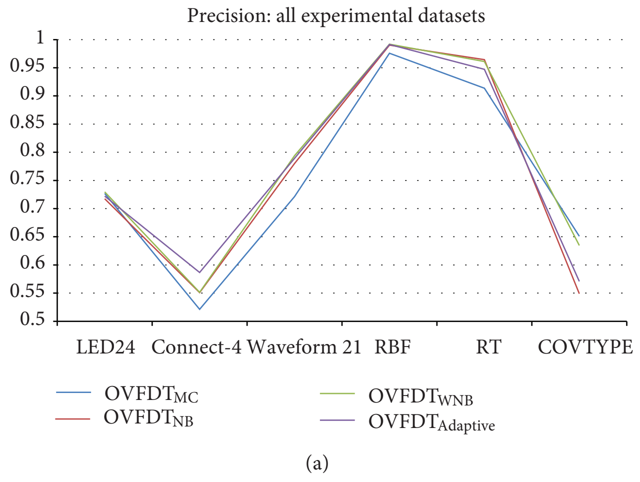
FIGURE 8: Tree size of the classification experiments by different FL types of OVFDT with datasets of imbalanced data class.

Another good performance benchmark is the receiver operating characteristic (ROC), which is a standard method for analyzing and comparing classifiers when the costs of misclassification are unknown. In a stream-mining scenario, it is not possible to know the misclassification costs, because the mining process is incremental over running data streams, and does not analyze a full dataset. The ROC provides a convenient graphical display of the tradeoff between the true and false positive classification rates for two class problems [11]. In the decision tree classification, however, there are more than two classes. Therefore, we extend the standard ROC model to a multiclass ROC analysis to evaluate the tree learning algorithm's performance.

Suppose that there is a D-class classification system, with d-dimensional classes that need to be classified by the tree learning algorithm. A $d \times d$-dimensional confusion matrix or contingency table C, which summarizes the results of the classifications, presents the true positives and false positives for the multi-class analysis. Each entry C_{ii} of the matrix C gives the number of examples, whose true class was A_i, that were actually assigned to A_i, where $1 \le i \le d$. Each entry C_{ij} of the matrix C gives the number of examples, whose true class was A_i, that were actually assigned to A_j, where $i \neq j$ and $1 \le i, j \le d$:

$$C = \begin{bmatrix} C_{11} & \cdots & C_{1d} \\ \vdots & \ddots & \vdots \\ C_{d1} & \cdots & C_{dd} \end{bmatrix}. \qquad (9)$$

To use two-class ROC statistics, each class i to d in the multi-class ROC is assigned a negative or positive value. Samples with class i are positive; otherwise, negative. True positives (TP) are examples correctly labeled as positives. False positives (FP) refer to negative examples incorrectly labeled as positive. True negatives (TN) are negatives correctly labeled as negative. Finally, false negatives (FN) refer to positive examples incorrectly labeled as negative. Each



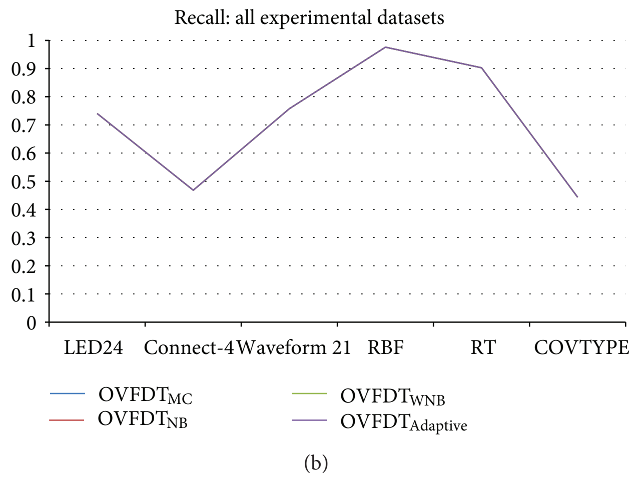


FIGURE 9: Precision and Recall values of the classification experiments by different FL types of OVFDT with datasets of imbalanced data class.

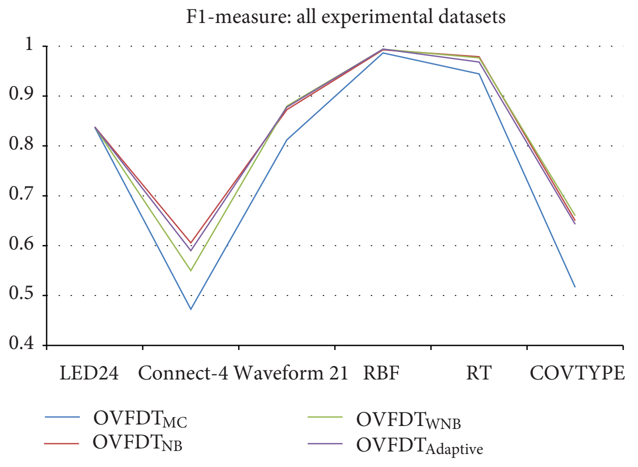


FIGURE 10: F1-measure of the classification experiments by different FL types of OVFDT with datasets of imbalanced data class.

class i can be converted into a two-class problem, with the corresponding values of *True Positive* (10), *False Positive* (11), *False Negative* (12), and *True Negative* (13) (see Figure 7):

$$\text{TP}_i = C_{ii}\ , \tag{10}$$

$$\text{FP}_i = \left(\sum_{j=1}^{d} C_{ji}\right) - C_{ii}, \tag{11}$$

$$\text{FN}_i = \left(\sum_{j=1}^{d} C_{ij}\right) - C_{ii}, \tag{12}$$

$$\text{TN}_i = \left(\sum_{i=1}^{d}\sum_{j=1}^{d} C_{ij}\right) - \text{TP}_i - \text{FP}_i - \text{FN}_i. \tag{13}$$

Precision-Recall is a well-known method of analyzing ROC. In pattern recognition, precision is the fraction of retrieved instances that are relevant, while recall is the fraction of relevant instances that are retrieved. The values of precision and recall range from 0 to 1. A precision score of 1 for a class i means that every item labeled as belonging to class i does indeed belong to class i. A recall score of 1 means that every item from class i was labeled as belonging to class i. Precision-Recall scores are not analyzed in isolation. F_β-measure [12] is a weighted harmonic mean of the Precision-Recall measure. The F1-measure evenly weights precision and recall scores. The best value for the F1-measure is 1 and the worst score is 0. In addition, the true positive rate (TPR) and the false positive rate (FPR) are common benchmarks in ROC analysis:

$$\begin{aligned}\text{Precision}_i &= \frac{\text{TP}_i}{(\text{TP}_i + \text{FP}_i)} = \frac{C_{ii}}{\left(\sum_{j=1}^{d} C_{ji}\right)},\\ \text{Recall}_i &= \frac{\text{TP}_i}{(\text{TP}_i + \text{FN}_i)} = \frac{C_{ii}}{\sum_{j=1}^{d} C_{ij}},\\ F_1\ \text{Measure}_i &= \frac{2\text{TP}_i}{\text{TP}_i + \text{FN}_i + \text{TP}_i + \text{FP}_i}\\ &= \frac{2C_{ii}}{\sum_{j=1}^{d} C_{ij} + \sum_{j=1}^{d} C_{ji}}.\end{aligned} \tag{14}$$

We analyze the Precision-Recall for each class for all the imbalanced class datasets. Due to limited space, the detailed charts are given in the Appendix. The average Precision-Recall values are described in Figures 9, and 10.

These charts illustrate that the average precision of OVFDT_{MC} is worse than those of the other methods. $\text{OVFDT}_{\text{Adaptive}}$ obtains the highest precision in homogenous (nominal only and numeric only) datasets. All methods have the same average values of recall (the lines appear to overlap). We then apply the F1-measure to evaluate the experiment result. As the chart below shows, the value ranges from 0 to 1; OVFDT_{MC} again has the lowest F1-measure value. However, because the datasets

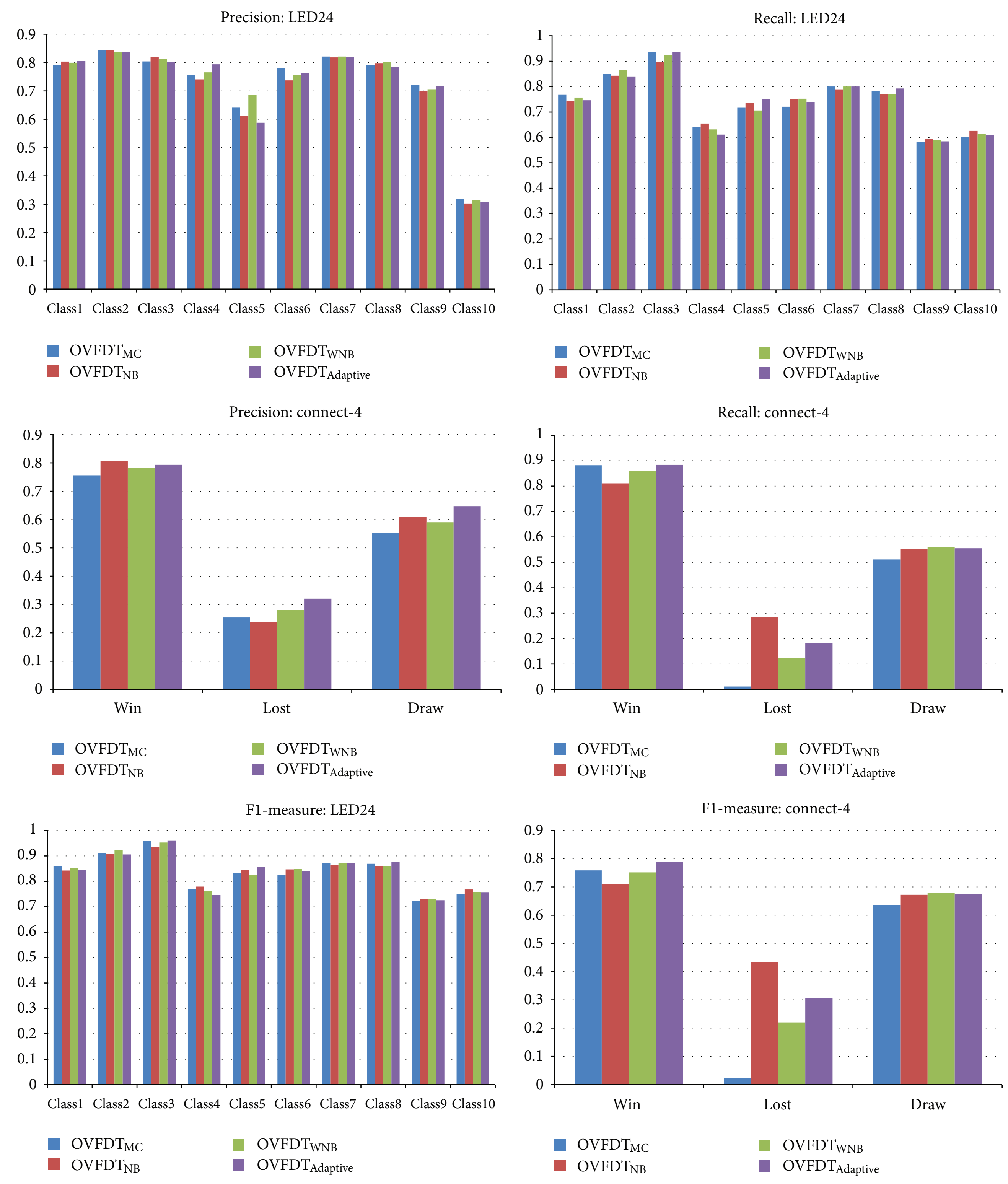


FIGURE 11

contained biased instances in imbalance classes, the average Precision-Recall analysis is not sufficient to determine accuracy. We must consider the Precision-Recall for the distributed class in every different data stream. From the above experiments, we observe that $\text{OVFDT}_{\text{Adaptive}}$ always achieves higher precision, recall, and F1-measure values than OVFDT_{MC}, but this was not the case for OVFDT_{NB} and $\text{OVFDT}_{\text{WNB}}$.

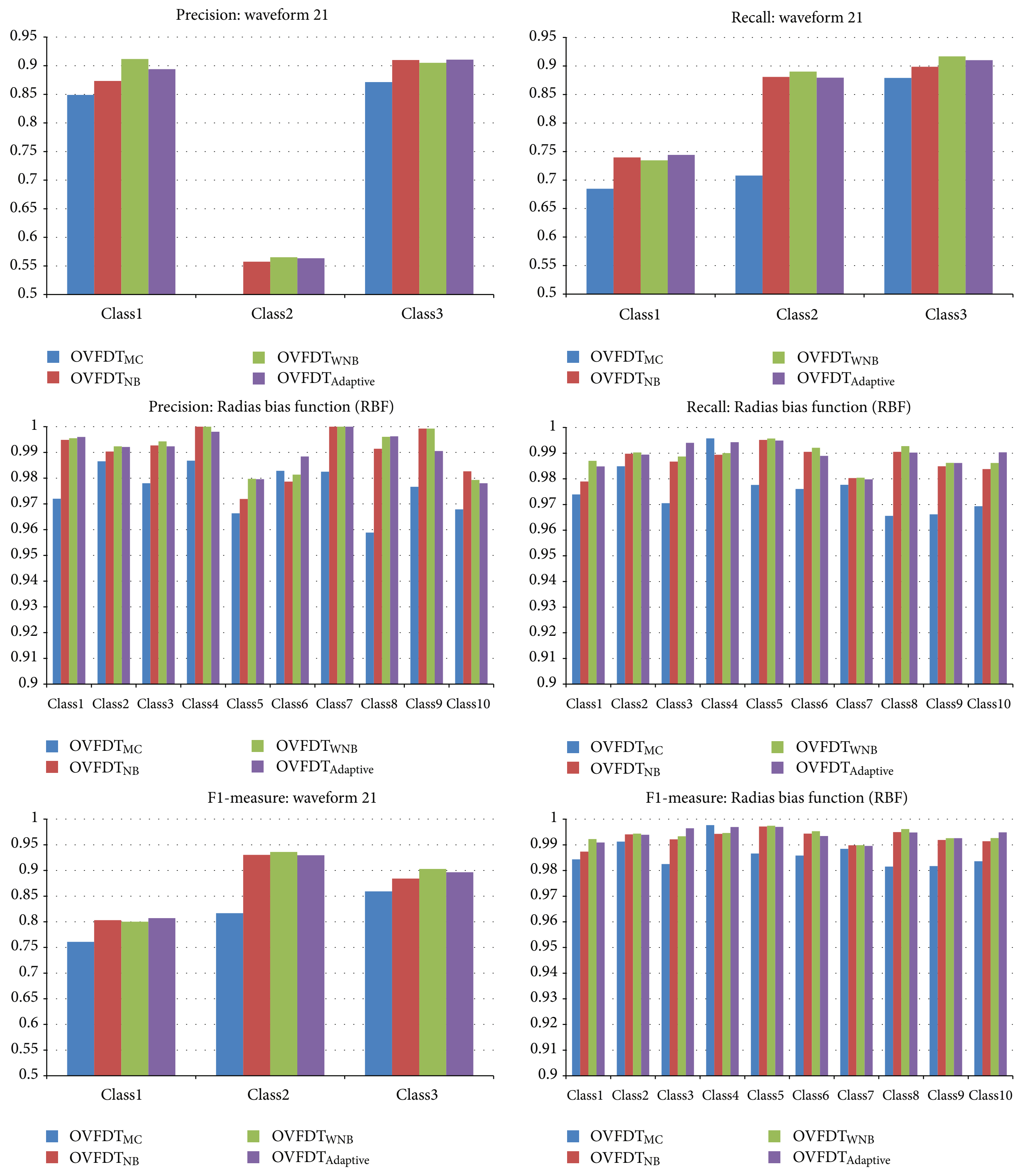


Figure 12

4. Conclusion

Imbalanced data classification is a challenging problem that generally refers to a learning model created for a dataset that has far more samples in one class than in the others. In an ubiquitous environment such as a wireless sensor network, it is not uncommon for the data of interest to fall into a small minority class. Previous researchers have tackled this problem by using techniques that inevitably create additional computation overheads. These techniques usually include

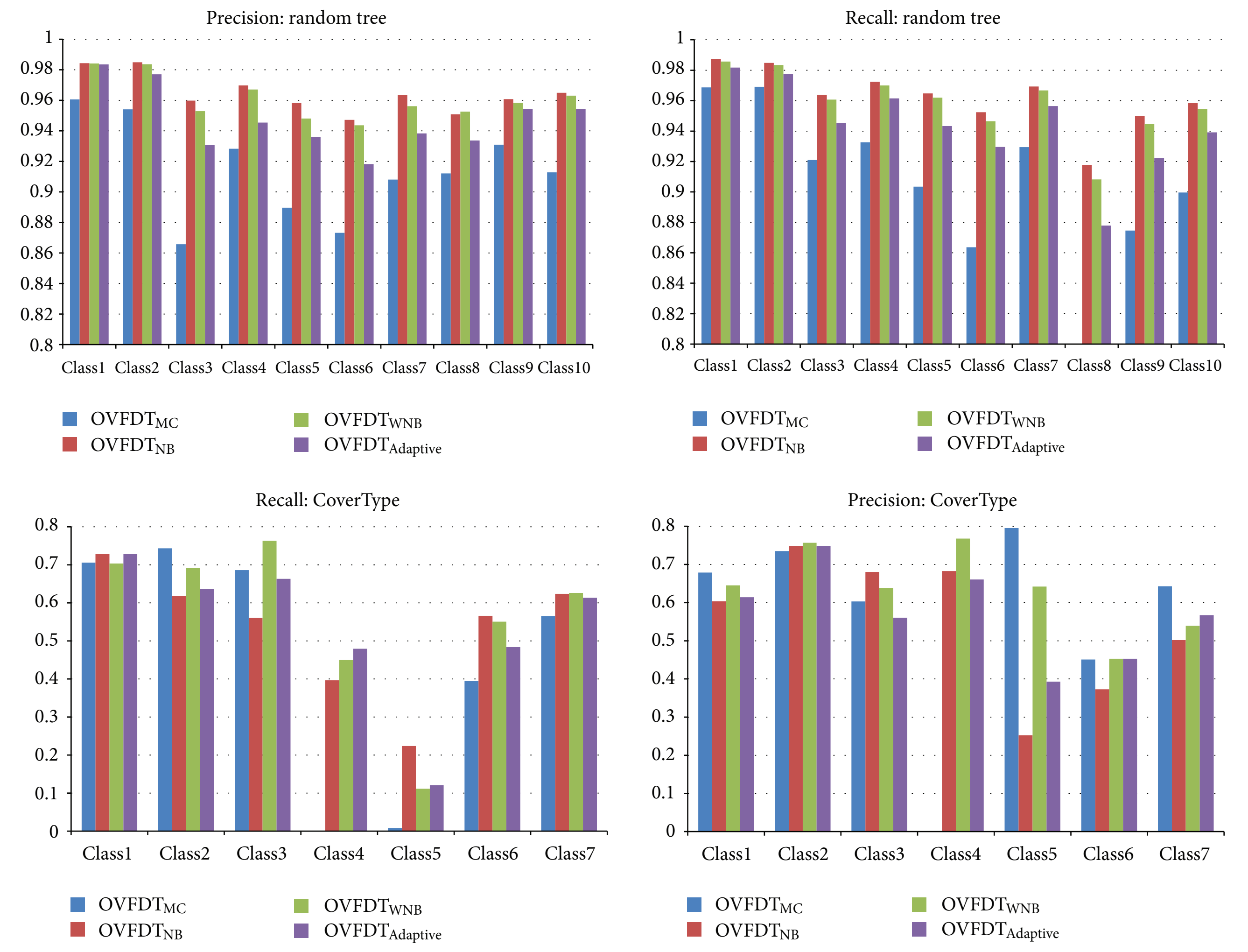


Figure 13

resampling the observations from a bounded archive so as to balance the imbalance. Others may resort to postpruning the decision tree and redistributing the classification costs in a backward-learning process. All of these proposed techniques worked well in traditional data mining but might not suit a real-time stream-mining scenario, where all the data arrive in a single pass; at a sensor sink it is neither practical nor feasible to archive a stationary set of data, let alone to resample.

In this paper, a novel solution is introduced at the algorithmic level, which is based on a popular stream-mining algorithm called the Very Fast Decision Tree (VFDT). Three modifications are proposed for VFDT as a means to reduce the effect of imbalanced class data. The modifications are implemented at the training phase prior to expanding the decision tree and at the testing phase, where prediction accuracy is fine-tuned by weighting the leaves of the decision trees according to the probabilities of the arriving data. The overall solution is called the Optimized VFDT with Functional Tree Leaf (OVFDT + FL). The mechanism of Function Tree Leaf is implemented by using weighted naïve Bayes predictors, installed at the decision tree leaves of the OVFDT. Specifically, perturbed datasets that include "biased" class distribution are used for experiments for illustrating the efficacy of the new algorithm. OVFDT + FL is shown to outperform VFDT in a series of experiments where datasets are deliberately biased by a custom-made data generator software program. In particular, two variants of FL called adaptive and weighted naïve Bayes performed consistently better than other techniques. OVFDT succeeded in minimizing the impacts of imbalanced class data, while maintaining high accuracy and a compact decision tree size. This contrasts with the known over-fitting problems of poor accuracy and huge tree size usually caused by imbalanced class data. The OVFDT + FL is validated as a good classification model for wireless sensor networks.

Appendix

A. Precision-Recall Charts for Each Dataset

A.1. Homogenous Data: Nominal Only. See Figure 11.

A.2. Homogenous Data: Numeric Only. See Figure 12.

A.3. Both Nominal and Numeric Attributes. See Figure 13.

Acknowledgments

The authors are thankful for the financial support from the research Grant "Real-time Data Stream Mining," Grant no. RG070/09-10S/FCC/FST, offered by the University of Macau, FST, and RDAO.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] M. Di and E. M. Joo, "A survey of machine learning in wireless sensor netoworks—from networking and application perspectives," in *Proceedings of the 6th International Conference on Information, Communications and Signal Processing (ICICS '07)*, pp. 1–5, Singapore, December 2007.

[3] Y. L. Borgne and G. Bontempi, "Round robin cycle for predictions in wireless sensor networks," in *Proceedings of the 2nd International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pp. 253–258, December 2005.

[4] J. Zhang, E. Bloedorn, L. Rosen, and D. Venese, "Learning rules from highly unbalanced data sets," in *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM '04)*, pp. 571–574, November 2004.

[5] J. Yu, Y. Qi, G. Wang, Q. Guo, and X. Gu, "An energy-aware distributed unequal clustering protocol for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2011, Article ID 202145, 8 pages, 2011.

[6] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," *International Journal of Knowledge Engineering and Soft Data Paradigms*, vol. 3, no. 1, pp. 4–21, 2011.

[7] J. Wang, M. Xu, H. Wang, and J. Zhang, "Classification of imbalanced data by using the SMOTE algorithm and locally linear embedding," in *Proceedings of the 8th International Conference on Signal Processing (ICSP '06)*, pp. 16–20, November 2006.

[8] Y. Zhai, N. Ma, B. An, and D. Ruan, "An effective over-sampling method for imbalanced data sets classification," *Chinese Journal of Electronics*, vol. 20, no. 3, pp. 489–494, 2011.

[9] P. Domingos, "MetaCost: a general method for making classifiers cost-sensitive," in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 155–164, San Diego, Calif, USA, 1999.

[10] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine Learning*, vol. 30, no. 2-3, pp. 195–215, 1998.

[11] Y. Hang, S. Fong, G. Sun, and R. Wong, "A very fast decision tree algorithm for real-time data mining of imperfect data streams in a distributed wireless sensor network," *International Journal of Distributed Sensor Networks*, vol. 2013, Article ID 863545, 2013.

[12] J. A. O. Gama, R. Rocha, and P. Medas, "Accurate decision trees for mining high-speed data streams," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, pp. 523–528, New York, NY, USA, August 2003.

[13] Y. Hang and S. Fong, "OVFDT with functional tree leaf—majority class, naive bayes and adaptive hybrid integrations," in *Proceedings of the 3rd International Conference on Data Mining and Intelligent Information Technology Applications (ICMIA '11)*, IEEE Press, Macau, China, October 2011.