Thai Journal of **Math**ematics
Volume 18 Number 4 (2020)
Pages 2041–2051

http://thaijmath.in.cmu.ac.th

ISSN 1686-0209

Recursive Tube-Partitioning Algorithm for a Class Imbalance Problem

Suebkul Kanchanasuk and Krung Sinapiromsaran*

Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, Bangkok 10330, Thailand
e-mail : suebkul.k@gmail.com (S. Kanchanasuk); krung.s@chula.ac.th (K. Sinapiromsaran)

Abstract A standard classifier acquired from a machine learning literature aims to categorize an instance into a well-defined class having comparable number of instances while data from real world problems tend to be imbalance. One way to deal with this imbalance problem is to modify the standard classification algorithm to capture minority instances and majority instances simultaneously. This work modifies the recursive partitioning algorithm based on a set of tubes, called the tube-tree algorithm. A tube-tree is a collection of tubes building from the combination of the input attributes where an internal node contains distinct class tubes corresponding to their respective classes. A tube composes of three components: a core vector, a tube length, and a tube radius built for each class regardless of its size suitable for an imbalance dataset. The forty six experiments are derived from the KEEL repository to compare the performance of the tube-tree with the support vector machine, the decision tree from C4.5, the decision tree from C5.0, and the naive Bayes classifier. The results of the tube-tree show the improvement over other classifiers of recall, and F1-measure except precision via the Wilcoxon signed rank test.

MSC: 68Q32; 68T20; 68T05
Keywords: tube-tree; recursive partitioning technique; machine learning; artificial intelligence

Submission date: 31.01.2018 / Acceptance date: 02.01.2019

1. Introduction

A class imbalance problem is an interesting issue in data mining [1, 2]. Although the amount of data at present is increasing, the number of instances for the required target class may not be sufficient. For instance, there are a large amount of collected patient data in a hospital, but data of rare disease patients may not be abundant enough to build an acceptable predictive model. A class imbalance problem is a classification problem with the highly different number of instances among classes. The minority class is defined as a class having relatively small number of instances, called minority instances and the other class having large number of instances, called majority instances [3]. A minority instance is normally labeled as a positive and a majority instance is labeled as a negative while the ratio between the total number of negatives over the total number of positives

*Corresponding author.

Published by The Mathematical Association of Thailand.
Copyright © 2020 by TJM. All rights reserved.

is defined as an imbalance ratio (IR). Since several classification algorithms are designed to maximize the number of correct class instances, this causes the high misclassification rate for minority instances. The class imbalance problems such as fault diagnosis [4], anomaly detection [5], and medical diagnosis [6, 7] use different techniques to deal with this imbalance nature. Three approaches for dealing with an imbalance problem are:

- **Pre-processing approach**. Pre-processing technique deals with imbalance by balancing class instances in a dataset before it feeds as the input for a classification algorithm. The re-sampling technique is applied by re-balancing the sample distribution: over-sampling or under-sampling or both. The over-sampling method is the method that synthesizes the new minority instances around known instances while the under-sampling method try to reduce majority instances according to specific criteria. The examples of oversampling technique are SMOTE [8], SMOTEBoost [9], Borderline-SMOTE [8], and Safe-Level SMOTE [3].
- **Cost sensitive learning approach**. This approach ascertains the appropriate costs for both positive and negative classes. To make a classifier recognizing a positive instance, the misclassification cost of a positive class instance is set to be higher than the misclassification cost of a negative class instance. The publications of this approach are shown in Sun et.al. in 2007 [10] generating the cost sensitive boosting for the class imbalance problems. Liu and Zhou in 2006 [11] applied the cost sensitive function technique to help neural network algorithm to determine the best weights for both positive and negative instances.
- **Algorithmic approach**. Some classifiers inherits a feature that can recognize class instances of any size, such as, a support vector machine. On the other hand, researchers attempted to modify a well-known classification algorithm to deal with imbalance such as the boosting technique[9, 12] or the entropy technique from Boonchuay and Sinapiromsaran in 2016 [13] and 2020 [14, 15].

In this paper, the recursive partitioning algorithm using a tube-tree has been proposed for imbalance as "TTIP". A tube-tree algorithm generates the series of tubes where a leaf node will be labeled by a single-class and an internal node will contain class tubes of the current dataset. The concept of a tube tree replaces the concept of the best entropy selection from the single attribute of a decision tree. Three components from each tube are computed from instances of each class which are a core vector, a tube length, and a tube radius. The core vector is the vector from the centroidal extreme instances to the centroid and the tube length is the longest magnitude of the vector of the centroidal extreme instances to all other instances, and the tube radius is the maximum of the shortest distance of any instances to the core vector. A tube is designed to capture the characteristic of all instances from each class and identify the boundary of the core vector by its tube length and its tube radius.

The next section covers literature review of classifiers. In section 3, the proposed method of a tube tree generation is given. In section 4, the results of the proposed algorithm are demonstrated. At the end, the conclusion and our future works are discussed.

2. Literature Review

Three standard classifiers that have been used for a class imbalance problem in literature are reviewed which are a naïve Bayes classifier, a support vector machine, and a decision tree.

2.1. Naïve Bayes

In particular, the naïve Bayes classifier [16] is a typical probabilistic classifier to be applied first. The naïve Bayes classifier adopts the statistical theory based on Bayes theorem which is described as

$$P(C|X_i) = \frac{P(X_i|C)P(C)}{P(X_i)} \tag{2.1}$$

where, $P(C|X_i)$ is the probability of the instance X_i being in the class C, $P(X_i|C)$ is the probability of having the instance X_i from the given class C, $P(C)$ is the probability of occurrence of class C, $P(D)$ is the probability of the instance X_i occurring.

2.2. Support Vector Machine

A support vector machine (SVM) is first introduced as a supervised learning classifier by Cortes and Vapnik in 1995 [17] for a binary classification. It is vastly used to solve a class imbalance problem appearing in many literatures. SVM is a learning kernel-based system that splits a high dimensional feature space using the optimal hyperplane.

2.3. Decision Tree

A decision tree consists of a set of nodes and edges where a node of decision tree is identified as an internal or a leaf node and the root node is just the first internal node. All internal nodes capture the attribute which will be branched by edges according to the result of the splitting criteria. Finally, a leaf node is the subset which labels the class of instances belong to this node.

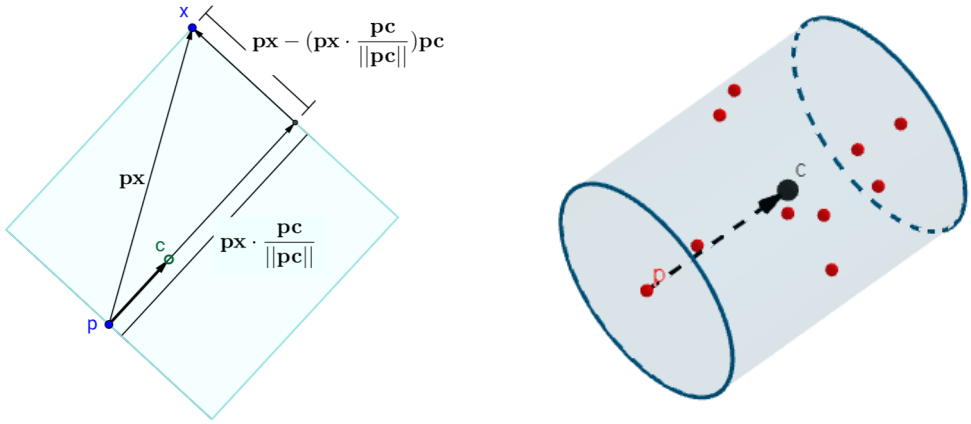
The most popular used decision tree algorithm during the past decade is C4.5 [18] which uses the gain ratio. The new and improved C4.5, called C5.0 applies several additional features, such as, creating a single classification tree, having the corresponding rule-based model, having C5.0's boosting procedure, and miscellaneous features of the algorithm [19].

3. Tube Tree for an Imbalance Problem (TTIP)

The first introduction of the tube tree algorithm, RBTP, appeared in 2016 by Kanchanasuk and Sinapiromsaran [20], named a recursive binary tube partitioning algorithm which applies to a general classification problem. It constructs the core vector using one of the extreme pole and the group centroid. Moreover, it recursively applies this tube construction on instances in an intersection of class tubes, until it reaches the terminal criteria. The result tube tree shows the statistical significant improvement over C4.5 and ctree [21].

To deal with a class imbalance problem, TTIP applies a different design of the core vector. The core vector is generated from the centroid and its furthest extreme instance instead of two extreme poles, called the centroidal extreme pole. For a binary class dataset, TTIP generates a tree-like structure for a given training dataset by recursively partitioning on the intersection of two "tubes". Each internal node including the root node contains two tubes for each class and the leaf node having only a single class will be

labeled with the target class. A tube is constructed from a core vector from the centroidal extreme pole while a tube length and a tube radius are constructed similar to RBTP.



(A) A tube covering $\mathbf{x}$ in 2D

(B) A tube for a single class dataset

FIGURE 1. Illustration of a tube where (A) a tube covers $\mathbf{x}$ with its formula in 2D, and (B) a tube covers the single class instances in the dataset using the maximum tube length and the tube radius.

The TTIP algorithm starts by generating all class tubes from a partition of class instances as shown in Algorithm 1. A tube of a single class consists of three components: a core vector ($\mathbf{pc}$) which is a vector with the direction starting from the centroidal extreme ($\mathbf{p}$) to the centroid ($\mathbf{c}$), and the magnitude of the core vector is the tube length (H), and the maximum extent along the core vector is the tube radius (R). The figure 1 shows the tube in 2D.

Algorithm 1 Generate a **Tube**($\mathbf{X}$)

Require: Subset $\mathbf{X}$ of size S having all N numeric attributes

1: Find the centroid ($\mathbf{c}$) where $\mathbf{c} = \frac{1}{S}\sum_{x \in X} \mathbf{x}$.

2: Find a centroidal extreme ($\mathbf{p}$) where $\mathbf{p} = \underset{\mathbf{x} \in \mathbf{X}}{\operatorname{argmax}} \{d(\mathbf{c}, \mathbf{x})\}$.

3: Compute a tube length: $H = \max_{\forall \mathbf{x} \in \mathbf{X}} \{\mathbf{h_x}\}$ where $\mathbf{h_x} = \frac{\mathbf{px} \cdot \mathbf{pc}}{\|\mathbf{pc}\|}$

4: Compute a tube radius: $R = \max_{\forall \mathbf{x} \in \mathbf{X}} \{\mathbf{r_x}\}$ where $\mathbf{r_x} = \left\| \mathbf{px} - \frac{\mathbf{px} \cdot \mathbf{pc}}{\|\mathbf{pc}\|} \mathbf{pc} \right\|$

Return: **Tube**($\mathbf{X}$) = ($\mathbf{pc}, H, R$)

Algorithm 2 Construct a Tube tree: **TubeNode**($\mathbf{X}$, **Target**)

Require: $\mathbf{X}$ is a subset with N numeric attributes and s is the corresponding target class s_1 or s_2.

1: Partition $\mathbf{X}$ according to the target class as $X_{s_i} = \{(\mathbf{x}, s)|s = s_i\}, i = 1, 2$.

if the size of $X_{s_i} < 3$ **then**

STOP. (Require more than 2 instances in the class s_i).

end if

2: Construct the tubes from X_{s_1} and X_{s_2}:
$\mathbf{Tube}(\mathbf{X}_{s_i}) = (\mathbf{pc}_{s_i}, H_{s_i}, R_{s_i})$ for $i = 1, 2$.

3: Compute $\mathbf{ApplyTube}(\mathbf{x}, \mathbf{Tube}(X_{s_i})) = (\mathbf{h_x} \leq H_{s_i}) \wedge (\mathbf{r_x} \leq R_{s_i})$.

4: Find $\mathbf{X}_{unknown} = \{\mathbf{x}| \bigwedge_{i\in\{1,2\}} \mathbf{ApplyTube}(\mathbf{x}, \mathbf{Tube}(X_{s_i})) = \text{True}\}$.

5: Recursively apply $\mathbf{TubeNode}(\mathbf{X}_{unknown})$.

Instances appearing in the single tube will be classified as the intended class while instances lie in the intersection of two tubes will be partitioned recursively by a tube tree algorithm as shown in Algorithm 2. The algorithm stops when all instances can be classified or the number of instances is too small to generate a reliable tube.

The frequently used performances for an imbalance problem are precision, recall, F1-measure, and G-measure where their corresponding formulae are described in [22].

4. Experimental Results

This section details the experimental results of TTIP from the KEEL dataset repository [23]. The collection of these datasets contains different number of imbalance ratios between 1.82 to 41.4. Table 1 shows the details of the datasets used in the experiments. The columns of Table 1 contain the reference number, name of the dataset, abbreviated name, the number of attributes, the number of instances in the positive class, the number of instances in the negative class, and imbalance ratio, respectively. All datasets are modified to have only two classes: positive and negative class. The second column of Table 1 shows the class of instances to be grouped as the positive class and the negative class separated by vs in the dataset name. For example, at the second row of Table 1, the dataset name "ecoli-0-1-4-6_vs_5" uses the target classes 0, 1, 4, 6 as the positive class and 5 as the negative class.

In this research, all codes, TTIP, C4.5, C5.0 and SVM [24] are implemented and run in R. C4.5 is called via the packages "J48" [25, 26], C5.0 is called via the package"C50" [27], Naive Bayes and SVM are called via the package "e1071" [28] using different kernel functions: linear, radial basis, sigmoid, and polynomial function.

To assess the quality of a classifier, the 5-fold cross-validation is applied on four performance measures: precision, recall, F1-measure, and G-measure (Fowlkes-Mallows index). The reported measures are generated from the average measures of all 5-fold cross-validation. Each dataset is repeated five times using different random sampling and is verified using the Wilcoxon signed-rank test to confirm the statistical significant.

TABLE 1. Dataset summary: The columns are the reference number(Ref.), the name of the dataset(dataset), the abbreviated name(Abbr), the number of the attributes(Att), the number of instances in the positive class(Pos), the number of instances in the negative class(Neg), and imbalance ratio(IR), respectively.

Ref.	Dataset	Abbr	Att	Pos	Neg	IR	Ref.	Dataset	Abbr	Att	Pos	Neg	IR
1	ecoli-0-1-3-7_vs_2-6	ecoli-a	7	7	274	39.14	24	glass-0-1-6_vs_5	glass-e	9	9	175	19.44
2	ecoli-0-1-4-6_vs_5	ecoli-b	6	20	260	13	25	glass-0-4_vs_5	glass-f	9	9	83	9.22
3	ecoli-0-1-4-7_vs_2-3-5-6	ecoli-c	7	29	307	10.59	26	glass-0-6_vs_5	glass-g	9	9	99	11
4	ecoli-0-1-4-7_vs_5-6	ecoli-d	6	25	307	12.28	27	glass0	glass-h	9	70	144	2.06
5	ecoli-0-1_vs_2-3-5	ecoli-e	7	24	220	9.17	28	glass1	glass-i	9	76	138	1.82
6	ecoli-0-1_vs_5	ecoli-f	6	20	220	11	29	glass2	glass-j	9	17	197	11.59
7	ecoli-0-2-3-4_vs_5	ecoli-g	7	20	182	9.1	30	glass4	glass-k	9	13	201	15.46
8	ecoli-0-2-6-7_vs_3-5	ecoli-h	7	22	202	9.18	31	glass5	glass-l	9	9	205	22.78
9	ecoli-0-3-4-6_vs_5	ecoli-i	7	20	185	9.25	32	glass6	glass-m	9	29	185	6.38
10	ecoli-0-3-4-7_vs_5-6	ecoli-j	7	25	232	9.28	33	yeast-0-2-5-6_vs_3-7-8-9	yeast-a	8	99	905	9.14
11	ecoli-0-3-4_vs_5	ecoli-k	7	20	180	9	34	yeast-0-2-5-7-9_vs_3-6-8	yeast-b	8	99	905	9.14
12	ecoli-0-4-6_vs_5	ecoli-l	6	20	183	9.15	35	yeast-0-3-5-9_vs_7-8	yeast-c	8	50	456	9.12
13	ecoli-0-6-7_vs_3-5	ecoli-m	7	22	200	9.09	36	yeast-0-5-6-7-9_vs_4	yeast-d	8	51	477	9.35
14	ecoli-0-6-7_vs_5	ecoli-n	6	20	200	10	37	yeast-1-2-8-9_vs_7	yeast-e	8	30	917	30.57
15	ecoli-0_vs_1	ecoli-o	7	77	143	1.86	38	yeast-1-4-5-8_vs_7	yeast-f	8	30	663	22.1
16	ecoli1	ecoli-p	7	77	259	3.36	39	yeast-1_vs_7	yeast-g	7	30	429	14.3
17	ecoli2	ecoli-q	7	52	284	5.46	40	yeast-2_vs_4	yeast-h	8	51	463	9.08
18	ecoli3	ecoli-r	7	35	301	8.6	41	yeast-2_vs_8	yeast-i	8	20	462	23.1
19	ecoli4	ecoli-s	7	20	316	15.8	42	yeast1	yeast-j	8	429	1055	2.46
20	glass-0-1-2-3_vs_4-5-6	glass-a	9	51	163	3.2	43	yeast3	yeast-k	8	163	1321	8.1
21	glass-0-1-4-6_vs_2	glass-b	9	17	188	11.06	44	yeast4	yeast-l	8	51	1433	28.1
22	glass-0-1-5_vs_2	glass-c	9	17	155	9.12	45	yeast5	yeast-m	8	44	1440	32.73
23	glass-0-1-6_vs_2	glass-d	9	17	175	10.29	46	yeast6	yeast-n	8	35	1449	41.4

TTIP is compared with four classifiers: two decision trees from C4.5 and C5.0, one statistical classification algorithm from the naive Bayes classifier, and four hyper-planes classification algorithms from the support vector machine with 4 kernel functions via precision, recall, F1-measure, and G-measure. To demystify which methods are appropriate, the non-parametric statistical hypothesis testing which was used in this comparison is Wilcoxon signed-rank test ($p < 0.05$). The symbols in the result are used to indicate the performance comparisons with the proposed algorithm where ▲ shows the significantly improvement of TTIP, △ shows no significantly improvement, and ▽ shows the lower performance.

4.1. TTIP vs. Decision Tree

The results of TTIP comparing with two decision trees from C4.5 and C5.0 are shown in Table 2. The average precision of TTIP improvement from the decision tree appears in 19 out of 46 datasets. The Wilcoxon signed rank test confirms that the precision of TTIP are statistically significantly improved over C4.5 in 13 datasets and C5.0 in 10 datasets. Basically, the average recall of TTIP in the fifth and twelfth columns outperform the decision tree over 25 datasets. It improved significantly over C4.5 in 15 datasets and for C5.0 in 12 datasets. Moreover, TTIP has the highest average F1-measure in 19 datasets. Moreover, the significantly improvement over C4.5 in 21 datasets and of C5.0 in 14 datasets is shown. Finally, there are 20 datasets which average G-measure of TTIP

is the largest but only 17 datasets is significantly improved over C4.5 and 13 datasets over C5.0.

Table 2. TTIP vs. Decision tree

Ref.	Dataset	IR	Precision		Recall		F1-measure		G-measure		Ref.	Dataset	IR	Precision		Recall		F1-measure		G-measure	
			C4.5	C5.0	C4.5	C5.0	C4.5	C5.0	C4.5	C5.0				C4.5	C5.0	C4.5	C5.0	C4.5	C5.0	C4.5	C5.0
1	ecoli-a	39.14	△	▲	▲	▲	▽	△	△	▲	24	glass-e	19.44	▽	▽	-	▽	▽	▽	▽	▽
2	ecoli-b	13.00	▲	▲	▲	△	▲	▲	▲	▲	25	glass-f	9.22	▽	-	▽	▽	▽	▽	▽	▽
3	ecoli-c	10.59	▽	▽	△	△	▽	▽	▽	▽	26	glass-g	11.00	▽	▽	▽	▽	▽	▽	▽	▽
4	ecoli-d	12.28	▽	▽	▲	△	▲	▽	▲	▽	27	glass-h	2.06	▽	▽	▽	▽	▽	▽	▽	▽
5	ecoli-e	9.17	▽	▽	▽	△	▽	▽	▽	▽	28	glass-i	1.82	▽	▽	▲	△	▲	▽	▲	▽
6	ecoli-f	11.00	▲	△	△	▽	▲	△	▲	△	29	glass-j	11.59	△	△	▲	▲	▲	▲	▲	▲
7	ecoli-g	9.10	△	▽	▲	△	▲	▽	▲	▽	30	glass-k	15.46	▲	▲	△	▲	▲	▲	▲	▲
8	ecoli-h	9.18	▽	▽	△	△	▽	▽	▽	▽	31	glass-l	22.78	▽	▽	▽	▽	▽	▽	▽	▽
9	ecoli-i	9.25	▲	△	△	△	▲	△	▲	△	32	glass-m	6.38	△	△	△	▽	△	▽	△	▽
10	ecoli-j	9.28	▽	▽	▲	△	△	▽	△	▽	33	yeast-a	9.14	▽	▽	△	▽	▲	▽	△	▽
11	ecoli-k	9.00	△	△	△	△	▲	△	▲	△	34	yeast-b	9.14	▽	▽	△	▽	▲	▽	△	▽
12	ecoli-l	9.15	▲	▲	▲	△	▲	▲	▲	▲	35	yeast-c	9.12	▽	▽	△	△	▲	▲	△	△
13	ecoli-m	9.09	▽	▽	▽	▽	▽	▽	▽	▽	36	yeast-d	9.35	▽	▽	△	▽	△	▽	△	▽
14	ecoli-n	10.00	▽	▽	▽	▽	▽	▽	▽	▽	37	yeast-e	30.57	▽	▽	▽	▽	▽	▽	▽	▽
15	ecoli-o	1.86	▽	▲	▽	▽	▽	▲	▽	▲	38	yeast-f	22.10	▲	▲	▲	▲	▲	▲	▲	▲
16	ecoli-p	3.36	▽	▽	△	▽	▽	▽	▽	▽	39	yeast-g	14.30	▽	▽	▽	▽	▽	▽	▽	▽
17	ecoli-q	5.46	▽	▽	▲	▲	▲	△	▲	△	40	yeast-h	9.08	▽	▽	▲	▽	▲	▽	▲	▽
18	ecoli-r	8.60	▽	▽	▲	▲	▲	▲	△	▲	41	yeast-i	23.10	▲	▲	▲	▲	▲	▲	▲	▲
19	ecoli-s	15.80	△	▽	▲	▲	▲	△	▲	△	42	yeast-j	2.46	▽	▽	△	▽	△	▽	△	▽
20	glass-a	3.20	△	△	△	△	△	▲	△	▲	43	yeast-k	8.10	▽	▽	▽	▽	▽	▽	▽	▽
21	glass-b	11.06	▲	▽	▲	▲	▲	▲	▲	△	44	yeast-l	28.10	▽	▽	△	▲	△	▲	△	▲
22	glass-c	9.12	▽	▽	△	△	△	△	△	△	45	yeast-m	32.73	▽	▽	▽	▽	▽	▽	▽	▽
23	glass-d	10.29	△	△	▲	▲	▲	▲	▲	▲	46	yeast-n	41.40	▽	△	△	▲	△	▲	△	▲

4.2. TTIP vs. Naïve Bayes

Table 3. TTIP vs. Naïve Bayes classifier

Ref.	Dataset	IR	Precision	Recall	F1-measure	G-measure	Ref.	Dataset	IR	Precision	Recall	F1-measure	G-measure
1	ecoli-a	39.14	▲	▽	△	▽	24	glass-e	19.44	▲	▽	▲	▲
2	ecoli-b	13	▲	▽	▲	▲	25	glass-f	9.22	▲	▽	▲	▲
3	ecoli-c	10.59	▽	▲	▲	▲	26	glass-g	11	▲	▽	▲	▲
4	ecoli-d	12.28	▽	▲	▲	△	27	glass-h	2.06	▲	▽	▲	△
5	ecoli-e	9.17	▲	▲	▲	▲	28	glass-i	1.82	▲	▽	▲	▽
6	ecoli-f	11	▲	△	▲	▲	29	glass-j	11.59	▲	▽	▲	▲
7	ecoli-g	9.1	▲	▽	▲	▲	30	glass-k	15.46	▲	▲	▲	▲
8	ecoli-h	9.18	▲	▲	▲	▲	31	glass-l	22.78	▲	▽	▲	▲
9	ecoli-i	9.25	▲	▽	▲	▲	32	glass-m	6.38	▲	▽	▲	△
10	ecoli-j	9.28	▲	▽	▲	▲	33	yeast-a	9.14	▽	▲	▲	▲
11	ecoli-k	9	▲	▽	▲	▲	34	yeast-b	9.14	▲	▲	▲	▲
12	ecoli-l	9.15	▲	▽	▲	▲	35	yeast-c	9.12	▲	△	▲	△
13	ecoli-m	9.09	▲	▲	▲	▲	36	yeast-d	9.35	▲	▲	▲	▲
14	ecoli-n	10	▲	△	△	△	37	yeast-e	30.57	▲	▽	▲	▽
15	ecoli-o	1.86	▽	▲	△	▽	38	yeast-f	22.1	▲	▽	△	▽
16	ecoli-p	3.36	▲	▽	▲	▲	39	yeast-g	14.3	△	▽	△	▽
17	ecoli-q	5.46	▲	▽	▲	▲	40	yeast-h	9.08	▲	▲	▲	▲
18	ecoli-r	8.6	▲	▽	▲	△	41	yeast-i	23.1	▲	▲	▲	▲
19	ecoli-s	15.8	▲	▽	▲	▲	42	yeast-j	2.46	▲	▽	▲	▽
20	glass-a	3.2	▲	△	▲	▲	43	yeast-k	8.1	▲	▲	▲	▲
21	glass-b	11.06	▲	▽	▲	△	44	yeast-l	28.1	▲	▲	▲	▲
22	glass-c	9.12	▲	▽	▲	▽	45	yeast-m	32.73	▲	▲	▲	▲
23	glass-d	10.29	▲	▽	▲	△	46	yeast-n	41.4	▲	▲	▲	▲

The results of TTIP comparing with two decision trees from C4.5 and C5.0 are shown in Table 3. There are 41, 20, 46, and 39 datasets which the average precision, recall,

F1-measure, and G-measure of TTIP are better than those of the naïve Bayes classifier, respectively. TTIP shows no significant improvement over the naïve Bayes classifier in 5 datasets via F1-measure.

4.3. TTIP vs. SVM

The SVM is applied using four kernel functions: linear, radial basis, sigmoid, and polynomial to compare against TTIP. The comparison results of TTIP and SVM are shown in Table 4. TTIP outperforms in 17, 26, 23, and 20 datasets of precision, recall, F-measure, and G-measure, respectively. Out of 17, only 10 datasets statistically improve in precision from all four kernel functions. Out of 26 datasets in Recall, 17 datasets exhibit statistical significant. Out of 23 datasets, only 18 datasets exhibit statistical significant in F1-measure and out of 20, only 14 datasets exhibit statistical significant in G-measure.

Table 4. TTIP vs. Support vector machine

Ref.	Dataset	IR	Precision SVM				Recall SVM				F1-measure SVM				G-measure SVM			
			Linear	RBF	Sigmoid	Poly	Linear	RBF	Sigmoid	Poly	Linear	RBF	Sigmoid	Poly	Linear	RBF	Sigmoid	Poly
1	ecoli-a	39.14	▽	▽	▲	▽	▽	-	▲	▽	▽	▽	▲	▽	▽	▽	▲	▽
2	ecoli-b	13	△	▽	▲	△	△	▽	▲	▽	▲	▽	▲	△	▲	▽	▲	△
3	ecoli-c	10.59	▽	▽	△	▽	△	▲	▲	▲	△	▲	▲	△	△	▲	▲	△
4	ecoli-d	12.28	△	▽	▲	▽	△	▽	▲	△	△	▽	▲	▽	△	▽	▲	▽
5	ecoli-e	9.17	▽	▽	△	▽	△	▲	▲	△	△	▲	▲	▽	△	▲	▲	▽
6	ecoli-f	11	△	▽	▲	▽	△	▽	▲	▽	△	▽	▲	▽	△	▽	▲	▽
7	ecoli-g	9.1	▽	△	▲	▽	△	▲	△	△	△	▲	▲	▽	△	▲	▲	▽
8	ecoli-h	9.18	▽	▽	▲	▽	△	▲	▲	▲	▽	△	▲	▽	▽	△	▲	▽
9	ecoli-i	9.25	△	▲	▲	▽	△	▲	△	▽	△	▲	▲	▽	△	▲	▲	▽
10	ecoli-j	9.28	▽	▽	▲	▽	△	▲	▲	▲	▽	△	▲	△	▽	△	▲	△
11	ecoli-k	9	△	▲	▲	▽	△	▲	△	△	△	▲	▲	▽	△	▲	▲	▽
12	ecoli-l	9.15	△	▽	▲	▽	△	▽	▲	△	△	▽	▲	▽	△	▽	▲	▽
13	ecoli-m	9.09	▽	▽	▲	▽	△	▲	▲	▽	▽	▽	▲	▽	▽	▽	▲	▽
14	ecoli-n	10	△	▽	▽	▽	▽	▽	▲	△	▽	▽	▽	▽	△	▽	▽	▽
15	ecoli-o	1.86	-	▽	▽	▽	▲	▲	▲	▲	△	▲	▲	▲	△	▲	▲	▲
16	ecoli-p	3.36	△	▽	△	▲	▲	▲	▲	▲	▲	▽	▲	▲	▲	▽	▲	▲
17	ecoli-q	5.46	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
18	ecoli-r	8.6	△	△	▲	△	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
19	ecoli-s	15.8	△	▽	▲	▽	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
20	glass-a	3.2	▲	△	▲	△	▲	▽	▲	▲	▲	△	▲	▲	▲	△	▲	▲
21	glass-b	11.06	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
22	glass-c	9.12	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
23	glass-d	10.29	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
24	glass-e	19.44	△	▲	▲	▲	△	▲	▲	▲	△	▲	▲	▲	△	▲	▲	▲
25	glass-f	9.22	△	▲	▲	▲	△	▲	▲	▲	△	▲	▲	▲	△	▲	▲	▲
26	glass-g	11	△	▽	▲	▲	▲	▲	▲	▲	▽	▽	▲	▲	▽	▽	▲	▲
27	glass-h	2.06	△	▽	▲	△	▲	△	▲	▲	▲	△	▲	▲	▲	△	▲	▲
28	glass-i	1.82	▲	▽	▲	△	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
29	glass-j	11.59	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
30	glass-k	15.46	▲	▽	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
31	glass-l	22.78	△	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
32	glass-m	6.38	△	▽	▲	▽	▲	▲	△	△	▲	△	△	▽	▲	▽	△	▽
33	yeast-a	9.14	△	▽	▲	▽	▲	▽	▲	▽	▲	△	▲	△	▲	▽	▲	▽
34	yeast-b	9.14	△	▽	▲	▽	▽	▽	▲	▽	▽	▽	▲	▽	▽	▽	▲	▽
35	yeast-c	9.12	▽	▽	▽	▽	▽	△	▲	△	▲	▲	▲	▲	▽	△	▲	△
36	yeast-d	9.35	△	▽	△	▽	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
37	yeast-e	30.57	▲	▲	△	▽	▲	▲	▲	▽	▲	▲	▲	▽	▲	▲	▲	▽
38	yeast-f	22.1	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
39	yeast-g	14.3	▲	▽	▲	▽	▲	▲	▲	▽	▲	▲	▲	▽	▲	▲	▲	▽
40	yeast-h	9.08	▽	▽	▲	▽	△	▲	▲	▲	▲	▲	▲	▲	△	▲	▲	▲
41	yeast-i	23.1	▽	▽	▽	▽	△	▲	▲	▲	△	▲	▲	▲	△	▲	▲	▲
42	yeast-j	2.46	△	▽	△	▽	▲	▽	▲	▲	▲	▲	▲	▲	▲	▽	▲	▲
43	yeast-k	8.1	△	▽	▲	▽	△	▽	▲	▲	△	▽	▲	▲	△	▽	▲	▲
44	yeast-l	28.1	▲	▲	△	△	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
45	yeast-m	32.73	△	▽	▲	▽	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲
46	yeast-n	41.4	▲	▽	▲	▽	▲	▲	▲	△	▲	▲	▲	▲	▲	▲	▲	△

4.4. Discussion

From the previous section, the performance measurements of TTIP are compared with the 3 types of classifiers; the decision tree (C4.5 and C5.0), the basic classifier (naïve Bayes), and the hyper-plane classification (SVM with kernel function). Numerical values

in Table 5 represent the average of each measure from all 46 datasets. The first column is the name of the measure, the second to the eighth column is the pairwise different averages between TTIP and C4.5, C5.0, Naive, and SVM with linear, radial basis, sigmoid, and polynomial kernel function, respectively. The negative value exhibits that the performance of TTIP is better than the performance of other classifiers. The last column shows the average of all measures in 46 datasets of TTIP.

TABLE 5. The average predictive measure of the classifiers from 46 imbalanced datasets.

Overall	**C4.5**	**C5.0**	**Naive**	**SVM.lin**	**SVM.rbf**	**SVM.sig**	**SVM.pol**	**TTIP**
Precision	4.11	4.62	**-25.10**	0.24	2.39	**-24.32**	2.10	62.17
Recall	**-6.13**	**-2.39**	**-0.44**	**-11.51**	**-13.17**	**-32.48**	**-17.24**	62.91
F1-measure	**-3.45**	**-0.41**	**-18.52**	**-8.20**	**-8.74**	**-29.81**	**-12.53**	60.94
G-measure	**-2.11**	0.57	**-17.58**	**-6.57**	**-5.87**	**-28.49**	**-8.65**	61.62

5. Conclusions

In this research, we proposed the recursive partitioning algorithm, called TTIP, which improved the performance of the classification algorithm in a binary-class imbalance problem. The collection of tubes is to determine the class regions and classify the member of the dataset into the correct class. TTIP shows statistically significant improvement compared with three traditional classifiers using precision, recall, F1-measure, and G-measure.

From the experimental results, TTIP shows a better performance over all other classifiers under Recall and F1-Measure but it fails in Precision. That means most positive instances are identified via TTIP however, it predicts too many positive instances than the tree from C4.5, C5.0, SVM with three kernel functions: linear, radial basis and polynomial. Despite the low precision, F1-measure which is the combination of recall and precision exhibits the superior performance of TTIP over other classifiers.

Due to the maximum tube length and the tube radius, TTIP creates wide decision boundary that includes too many negative instances. The reshaped of the tube could alleviate the effect of those negative instances.

Acknowledgements

I would like to thank my advisor for the support, the referee for the comments and suggestions on the manuscript, and the Applied Mathematics and Computational Science Laboratory for the materials and tools.

References

[1] N.V. Chawla, N. Japkowicz, A. Kotcz, Editorial: special issue on learning from imbalanced data sets, ACM Sigkdd Explorations Newsletter 6 (1) (2004) 1–6.

[2] N.V. Chawla, Data mining for imbalanced datasets: An overview, Data mining and knowledge discovery handbook, Springer (2005), 853–867.

[3] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, Advances in knowledge discovery and data mining (2009) 475–482.

[4] Q. Jiang, M. Jia, J. Hu, F. Xu, Machinery fault diagnosis using supervised manifold learning, Mechanical Systems and Signal Processing 23 (7) (2009) 2301–2311.

[5] G. Giacinto, F. Roli, L. Didaci, Fusion of multiple classifiers for intrusion detection in computer networks, Pattern recognition letters 24 (12) (2003) 1795–1803.

[6] L. Mena, J.A. Gonzalez, Machine learning for imbalanced datasets: application in medical diagnostic, Flairs Conference (2006) 574–579.

[7] B. Krawczyk, Ł. Jeleń, A. Krzyżak, T. Fevens, Oversampling methods for classification of imbalanced breast cancer malignancy data, Computer Vision and Graphics (2012) 483–490.

[8] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, Journal of artificial intelligence research 16 (2002) 321–357.

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

[10] Y. Sun, M.S. Kamel, A.K.C. Wong, Y. Wang, Cost-sensitive boosting for classification of imbalanced data, Pattern Recognition 40 (12) (2007) 3358–3378.

[11] Z.H. Zhou, X.Y. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Transactions on Knowledge and Data Engineering 18 (1) (2006) 63–77.

[12] C. Seiffert, T.M. Khoshgoftaar, J. Van Hulse, A. Napolitano, RUSBoost: A hybrid approach to alleviating class imbalance, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 40 (1) (2010) 185–197.

[13] K. Boonchuay, K. Sinapiromsaran, C. Lursinsap, Decision tree induction based on minority entropy for the class imbalance problem, Pattern Analysis and Applications (2016) 1–14.

[14] A. Sagoolmuang, K. Sinapiromsaran, Decision tree algorithm with class overlapping-balancing entropy for class imbalanced problem, International Journal of Machine Learning and Computing 10 (3) (2020) 444–451.

[15] A. Sagoolmuang, K. Sinapiromsaran, Oblique decision tree algorithm with minority condensation for class imbalanced problem, Engineering Journal 24 (1) (2020) 221–237.

[16] P. Langley, S. Sage, Induction of selective Bayesian classifiers, Proceedings of the Tenth international conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc. (1994), 399–406.

[17] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (3) (1995) 273–297.

[18] J.R. Quinlan, C4.5: Programming for Machine Learning, Morgan Kauffmann Publishers Inc., San Francisco, 1993.

[19] M. Kuhn, K. Johnson, Applied Predictive Modeling 810, Springer, New York, 2013.

[20] S. Kanchanasuk, K. Sinapiromsaran, Recursive binary tube partitioning for classification, Intelligent and Evolutionary Systems (2016) 99–107.

[21] T. Hothorn, K. Hornik, A. Zeileis, Unbiased recursive partitioning: A conditional inference framework, Journal of Computational and Graphical statistics 15 (3) (2006) 651–674.

[22] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Transactions on knowledge and data engineering 21 (9) (2009) 1263–1284.

[23] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, Journal of Multiple-Valued Logic & Soft Computing 17 (2011).

[24] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2016.

[25] K. Hornik, C. Buchta, A. Zeileis, Open-source machine learning: R meets weka, Computational Statistics 24 (2) (2009) 225–232.

[26] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, 2nd. San Francisco: Morgan Kaufmann, 2005.

[27] M. Kuhn, S. Weston, M. Culp, N. Coulter, R. Quinlan, Package C50, (2020) https://cran.r-project.org/web/packages/C50/index.html.

[28] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, F. Leisch, C.C. Chang, C.C. Lin, M.D. Meyer, Package e1071, The R Journal (2019).