

ป่าสุ่มของต้นไม้ตัดสิ้นใจผสมและต้นไม้ตัดสิ้นใจควบแน่นฝายข้างน้อย
สำหรับปัญหาคลาสไม่ดุล



นางสาวสุพรรณ หอมจันทร์ดี

จุฬาลงกรณ์มหาวิทยาลัย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2565

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

RANDOM FOREST OF MIXED DECISION TREES AND MINORITY
CONDENSATION DECISION TREES FOR CLASS IMBALANCED PROBLEM

Miss. Suvaporn Homjandee



A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Applied Mathematics and

Computational Science

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2022

Copyright of Chulalongkorn University

Thesis Title RANDOM FOREST OF MIXED DECISION TREES AND
MINORITY CONDENSATION DECISION TREES FOR
CLASS IMBALANCED PROBLEM
By Miss. Suvaporn Homjandee
Field of Study Applied Mathematics and Computational Science
Thesis Advisor Associate Professor Krung Sinapiromsaran, Ph.D.

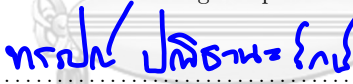
Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment
of the Requirements for the Master's Degree


..... Dean of the Faculty of Science
(Professor Polkit Sangvanich, Ph.D.)

THESIS COMMITTEE


..... Chairman
(Assistant Professor Kitiporn Plaimas, Ph.D.)


..... Thesis Advisor
(Associate Professor Krung Sinapiromsaran, Ph.D.)


..... Examiner
(Thap Panitanarak, Ph.D.)


..... External Examiner
(Assistant Professor Chumphol Bunkhumpornpat, Ph.D.)

สุพร หอมจันทร์ดี : ป่าสุ่มของต้นไม้ตัดสินใจผสมและต้นไม้ตัดสินใจควบนั่นฝ่ายข้างน้อยสำหรับปัญหาคลาสไม่ดุล. (RANDOM FOREST OF MIXED DECISION TREES AND MINORITY CONDENSATION DECISION TREES FOR CLASS IMBALANCED PROBLEM) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ.ดร. ดร.กรุง สีนอภิรมณ์สรายุ, 47 หน้า.

ในช่วงหลายปีที่ผ่านมาการใช้ข้อมูลในอดีตมาช่วยในการสร้างตัวแบบที่สามารถจำแนกประเภทได้อย่างมีประสิทธิภาพได้เข้ามามีบทบาทสำคัญในการเรียนรู้ของเครื่องคอมพิวเตอร์ แต่อย่างไรก็ตาม อัลกอริทึมการจำแนกประเภทมาตรฐาน จะประสบกับปัญหาในการจำแนกประเภท เมื่อต้องเผชิญหน้ากับข้อมูลที่ไม่สมดุลระหว่างคลาส วิทยานิพนธ์นี้จึงพัฒนาอัลกอริทึมสำหรับแก้ปัญหาเกี่ยวกับข้อมูลที่ไม่สมดุลที่มีคลาสสองชนิด โดยไม่ต้องปรับเปลี่ยนการกระจายตัวของข้อมูล โดยการสร้างตัวจำแนกประเภทที่ผสมผสานระหว่างต้นไม้ที่ใช้เอนโทรปีควบนั่นฝ่ายข้างน้อย (Minority Condensation Entropy) กับต้นไม้ตัดสินใจดั้งเดิม ให้เป็นป่าสุ่ม (Random Forest) ในการทดสอบจะนำชุดข้อมูลที่สังเคราะห์ขึ้นมา และชุดข้อมูลในชีวิตจริง 10 ชุด จากแหล่งเก็บข้อมูล UCI ในการทดสอบการแยกประเภท ผลเฉลี่ยมีค่า Precision, Recall, F-measure และ G-measure เป็น 0.6105, 0.7784, 0.6694, และ 0.6814 ตามลำดับ จะเห็นว่าป่าสุ่มที่ถูกพัฒนาขึ้นมาให้ประสิทธิภาพดีกว่าตัวแยกประเภทตัวดั้งเดิมและอัลกอริทึมอื่นๆ ดังนั้น ขั้นตอนวิธีที่พัฒนาขึ้นมาจึงมีประโยชน์ที่ช่วยการสร้างตัวแบบจำแนกประเภทสำหรับการไม่สมดุลคลาสมีประสิทธิผลมากขึ้น โดยไม่ต้องพิจารณาการกระจายตัวของข้อมูลและยังสามารถประยุกต์กับข้อมูลอื่นๆ ได้อีกด้วย

จุฬาลงกรณ์มหาวิทยาลัย

CHULALONGKORN UNIVERSITY

ภาควิชา	คณิตศาสตร์และ	ลายมือชื่อนิสิต	<i>อนุส วัฒนศิริ</i>
	วิทยาการคอมพิวเตอร์	ลายมือชื่อ อ.ที่ปรึกษาหลัก	<i>[Signature]</i>
สาขาวิชา	คณิตศาสตร์ประยุกต์	ลายมือชื่อ อ.ที่ปรึกษาร่วม	
	และวิทยาการคณนา		
ปีการศึกษา	2565		

6370037823 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE
 KEYWORDS : CLASS IMBALANCED, CLASSIFICATION, MINORITY CONDENSATION
 ENTROPY, RANDOM FOREST

SUVAPORN HOMJANDEE : RANDOM FOREST OF MIXED DECISION TREES
 AND MINORITY CONDENSATION DECISION TREES FOR CLASS IMBALANCED
 PROBLEM. ADVISOR : ASSOC. PROF. KRUNG SINAPIROMSARAN, Ph.D., 47 pp.

Building an effective classifier that could classify classes of instances in a dataset from historical data played an important role in machine learning for the past several years. However, a standard classification algorithm has difficulty generating an appropriate classifier when faced with imbalanced datasets. This research develops an algorithmic approach for handling a binary-class imbalanced problem directly from the original dataset by building a random forest using a mixture of standard decision trees and minority condensation decision trees (MCDT). The algorithm constructs MCDT from a bootstrapped dataset and a decision tree from a balanced bootstrapped dataset. Furthermore, the experimental results on synthetic datasets and ten real-world datasets from the UCI repository show that the proposed algorithm yields average Precision, Recall, F-measure, and G-measure as 0.6105, 0.7784, 0.6694, and 0.6814, respectively. Thus, the constructed algorithm outperforms the standard random forest and other algorithms. Therefore, the developed algorithm is useful in building a classifier for class imbalanced with higher performance without considering the distribution of the data and this technique can be adaptable to various datasets.

จุฬาลงกรณ์มหาวิทยาลัย
 CHULALONGKORN UNIVERSITY

Department : Mathematics and Student's Signature
 Computer Science Advisor's Signature
 Field of Study : Applied Mathematics and Co-advisor's Signature
 Computational Science
 Academic Year : 2022

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to my thesis advisor, Associate Professor Krung Sinapiroms, Ph.D., for his continuous supports in my studies for the master's degree. His encouragement, motivation, and guidance helped me during the time for conducting this research and writing this thesis until it was accomplished. I further would like to thank all of my dissertation committees: Assistant Professor Kitiporn Plaimas, Ph.D., Thap Panitana, Ph.D., and Assistant Professor Chumphol Bunkhumpornpat, Ph.D., for their insightful comments and suggestions which motivated me to extend my research from various perspectives.

Moreover, I would like to express my special appreciation and thanks to my financial sponsors, "Development and Promotion of Science and Technology Talents Project (DPST)" for the scholarship. My sincere thanks also go to the Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, which provided me a great opportunity to attend international conferences and facilities that I received throughout my graduate studies.

Finally, I would like to thank my family for supporting me throughout working and writing this thesis. Furthermore, I wish to express my gratitude to all my friends and colleagues who stayed with me and provided their encouragement, relaxation, great suggestions, and supports in many ways during a hard time studying for my master's degree.

CONTENTS

	Page
ABSTRACT IN THAI	iv
ABSTRACT IN ENGLISH	v
ACKNOWLEDGEMENTS	vi
CONTENTS	vii
CHAPTER	
1 INTRODUCTION	1
1.1 Introduction and literature surveys	1
1.2 Classification	4
1.3 Research Objective	6
2 Background knowledge	8
2.1 Classification Process Overview	8
2.2 Training Dataset Used For Classification	11
2.3 Decision tree	11
2.3.1 Decision Tree Induction Algorithm	14
2.3.2 Attribute Selection Measures	16
2.4 Random forests	18
2.4.1 Bootstrap Aggregation (Bagging)	18
2.4.2 Random Forest Model	20
2.5 Class imbalanced Problem	21
2.5.1 Class Imbalance Classification Challenging	21
2.5.2 Binary-Class Imbalanced Problem	22
3 Research Methodology	26
3.1 Random forest of mixed Minority condensation Decision Tree	26
3.2 Proposed Algorithm	27
4 Experiments and Results	32
4.1 Accuracy Performance	32
4.2 Experiment Results and discussion	35
4.2.1 Synthetic dataset	35

CHAPTER	Page
4.2.2 Real-world dataset	37
5 Conclusion	40
REFERENCES	42
BIOGRAPHY	47



CHAPTER I

INTRODUCTION

This chapter covers machine learning, classification, and research objectives. The first section is an introduction to machine learning. The second section covers classification in machine learning and the motivation of a class-imbalance problem and currently related works. The third section is the research objectives and the overview of this thesis.

1.1 Introduction and literature surveys

In 1956, the keyword “Artificial Intelligence (AI)” was mentioned for the first time in the Dartmouth summer research project [1] which was organized by Marvin Minsky, John McCarthy, Claude Shannon, and Nathan Rochester. The conference is the workshop for researchers interested in this field and related fields such as automata theory, neural networks, and intelligence studies. The definitions of AI are broadly defined. Later, it involves a human-led thinking and the reasoning process which are extended into several branches such as natural language processing, expert system, computer vision, robotics. Moreover, at present, the popular name, Machine Learning (ML), is used to identify this field. Later, deep learning is developed and is used to learn digital data from its binary representation. Figure 1.1 shows the relationship between AI, ML, and DL.

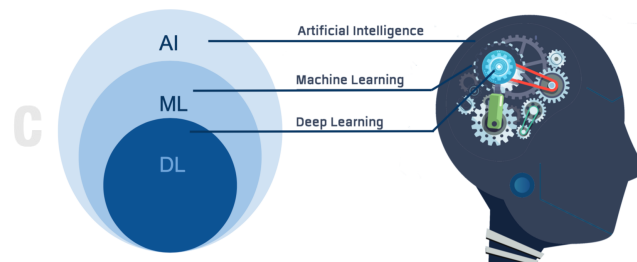


Figure 1.1: Relationship between AI and ML

Machine learning [2] is one of AI tasks focusing on using samples or experiences to

learn a well-defined task, where human involvement is only in the design of the system. The system will extract the essence of the task from training samples by itself. After learning is completed with training samples, this machine can be effectively applied to new samples that have never been seen before. The accuracy of this machine can be improved using more examples or experiences. Learning design can be applied to a wide variety of tasks. For example, in Playing chess [3], the task is to play chess to win where a performance value is calculated by the ratio of the number of wins and the total number of times played. The experience to be practiced is a record of human chess or obtained by the system played.

“Machine learning incorporates several hundred statistical-based algorithms and choosing the right algorithm(s) for the job is a constant challenge of working in this field. Before examining specific algorithms, it’s important to consolidate one’s understanding of the three overarching categories of machine learning and their treatment of input and output variables .”

- Machine Learning For Absolute Beginners (2017)[4] -

Moreover, ML is divided into categories based on the output results. It can be separated into three main categories:

Supervised Learning

Supervised learning is the task in machine learning to decode the relationship between input factors (independent variables) and their known outputs (dependent variables) using a labeled dataset. Variables that apparently influence the dependent variable (written as a lowercase “y”) are known as independent variables represented by the vector “ \mathbf{x} ”. For example, the temperature variable impacts the output of going for a run ($y = \text{“yes”}$). We can predict the decision of going outside for running by analyzing the relationship between the factor attributes such as Outlook, Temperature, Humidity, and the decision of the runner ($y = \text{“yes”}$ or $y = \text{“no”}$). Given that the supervised learning algorithm knows the final decision, it can work backward to determine the relationship between a runner’s decision (output) and its characteristics (input).

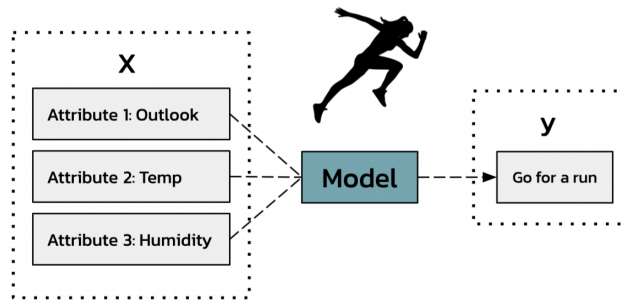


Figure 1.2: Runner predictive model

The machine builds a model that produces an outcome with new data based on the underlying patterns and rules learned from the training data when it deciphers the rules and patterns between X and y where matrix X contains a row of instances. After the model is developed and tested, it can be applied to test data and being evaluated for accuracy. Also, supervised learning is commonly solved (1) classification, such as image classification, identity fraud detection, customer retention, and diagnostics, and (2) regression, for example, population growth prediction, market forecasting, weather forecasting, advertising popularity prediction, and estimating life expectancy.

Unsupervised Learning

On the other hand, in unsupervised learning, there is no known output variable. Work focuses on finding the relationships between input variables and uncovering hidden patterns that can be extracted to build new outputs or new labels which are in line with those input variables. Unsupervised learning is commonly used in the clustering task such as the customer segmentation, the recommender system and the targeted marketing. Another use is to study relationships between buying items from customers' transaction data such as association rule analysis [5]. In addition, the dimensionality reduction is also commonly used, for example, a feature selection which is useful method for selecting the input data's optimal, relevant feature and removing irrelevant features in machine learning algorithm.

Reinforcement Learning

The third of machine learning is reinforcement learning, which works by gathering information during a sequence of interactions and gaining feedback from random trial and error. The concept can be best described through the use of a video game analogy. As a player goes through a game's virtual area, they learn the significance of various actions in various situations and get more familiar with the playing field. Thus, reinforcement learning is used in the field of game AI, real-time decision, robot navigation, learning tasks, and skill acquisition.

1.2 Classification

This research deals with **the classification of an imbalanced dataset** in supervised learning. It aims to build a classifier that could classify a target class in a dataset from historical data. There are many well-known classification models along with their algorithms that have been presented using various concepts. For example, the k-nearest neighbors classifier (KNN) [6] employs the concept of similarity, the Naïve Bayes model [7] and the logistic regression [8] rely on the knowledge of probability and statistics, a decision tree [9] and decision rules [10] employ the recursive partitioning method based on properties of a single attribute, a random forest [11] is a collection of small decision trees with sampling features that could be used to categorize instances, a support vector machine (SVM) [12] is related to linear separation, and an artificial neural network (ANN) [13] and deep learning [14] imitate the human brain's function.

Generating an appropriate classifier when they are faced with the complex information of imbalanced datasets or a dataset that has an extremely unequal class distribution is hard. Class imbalanced problems can be encountered in a variety of real-world situations, where minority classes are usually more concerned about correctly classifying than majority classes. For example, there are a tiny number of patients in disease diagnosis [15][16][17] compared to normal persons, but they are significant and must be detected as Figure 1.3 shows an imbalanced medical database, the tiny number of cancer patients (red color) compared with the number of normal persons (blue color) leads to a classifier

facing an extremely unequal dataset. Furthermore, the prediction of fraudulent transactions is more concentrated than non-fraudulent situations in fraud detection [18][19]. An e-mail spam filtering [20], protein/DNA identification [21], target detection [22], and text mining [23] are all examples of the class imbalanced problem. As a consequence, it is necessary to approach this problem carefully.

However, there are many methods developed for handling a binary-class imbalanced problem. They can be categorized into four different approaches [24] that are (1) a data-level approach such as SMOTE [25] algorithm carries out an oversampling approach to rebalance the original training set, or RFWs [26] (2) an algorithmic-level approach, many of popular machine learning algorithms have been subject to direct modifications of learners, including SVMs [27], decision trees [28], OMCT [29], NN approaches [30], Bayesian classifiers [31], or kernel machines [32] (3) a cost-sensitive approach such as ANNs [33], and (4) AdaBoost [34] and Boosting [35] are commonly considered in an ensemble approach for imbalanced problems. The research of an algorithm-level approach is of particular interest because it does not create any changes in the data distribution and is more flexible to varied characteristics of imbalanced datasets. Recently, in 2019, a self-balancing recursive partitioning algorithm for classification, the new splitting measure called the

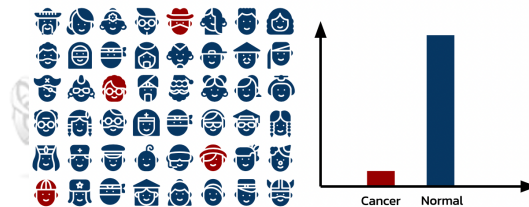


Figure 1.3 Example of an imbalanced cancer medical database

minority condensation entropy (MCE) [36] is also presented to handle an imbalanced dataset during the construction of a decision tree. The origin idea of it came from the splitting measure named the minority entropy (ME) [37], the condition that discards the majority instances that do not affect the partition from the minority class, and the interquartile range (IQR) rule to scope the range of minority values. The entropy computed with the set of instances within the minority range is assigned as MCE, and a decision

tree built based on MCE is called the minority condensation decision tree (MCDT).

Furthermore, random forest, one of the classifiers has received wide attention for adopting the imbalanced concept. It is the combination of the decision trees. the random forest algorithm contains two important steps that are (1) a bootstrap on a training set and (2) a randomization of attributes to build different trees. Nevertheless, when the bootstrap is used on an imbalanced dataset, there is a chance that most minorities will not be picked during the bootstrapping step. These reasons make the random forest face the hassle of constructing a non-bias model from an imbalanced dataset and are sensible to bootstrap only negative instances (majorities) and keep all positive instances (minorities).

These lead to the research that will augment decision trees built from MCDT on training with imbalance training datasets and the standard random forest on balance ones. The workflow of an algorithm in this research consists of 2 parts: (1) In a bootstrapping step of a training set, it will bootstrap only majorities and keep all minorities ignoring the balance of these two classes. Next, it will use MCE as the splitting condition to find the best attribute to split in each tree, (2) Additional bootstrap keeping all minorities and bootstrap only majorities until the number of majorities and minorities are the same. Next, it will use the entropy to construct the decision tree. Moreover, the efficiency of a classifier in a class imbalanced problem is evaluated quantitatively based on the precision, the recall, the F-measure, and the Geometric mean which are derived from the confusion matrix.

1.3 Research Objective

The objective of this research is to construct a random forest to be able to classify binary-class imbalanced datasets dealing with numeric attributes, called Random forest of mixed minority Condensation and Decision tree (RMCD) which extends from minority condensation decision trees. The algorithm is implemented, and experimented on the real world datasets and synthetic datasets, then it compares with other methods using the precision, the recall, the F1-measure and the

Geometric measure as their performance measures.

The remainder of this thesis is organized as follows. In Chapter II, some necessary background knowledge is explained. The formal definition of RMCD, as well as its algorithm and thresholds, are presented in Chapter III. Next, the experimental results show the accuracy performance of RMCD and Chapter IV provides the discussion and conclusion of this work.



CHAPTER II

BACKGROUND KNOWLEDGE

This chapter covers a classification process overview, the structure of a dataset that is used in the research, a decision tree induction, a classifier that extends using a collection of decision trees named as a random forest, an explanation of a class imbalanced problem, and random forest for an imbalanced dataset, as well as other background knowledge required to understand this dissertation. In addition, this chapter also contains a review of other related works.

2.1 Classification Process Overview

Classification is a systematic process of designing a model which is accurate enough to be believed [38][39][40]. As Figure 2.1 shows, the classification process required two data types: Training Data and Test Data. A training dataset is a collection of all the records that describe the past where these records contain information, including the categorical attributes for which a classifier is being designed. A test dataset is made up of all records that describe the present and for which we want to forecast the future, in which there are no values for category fields in these records. Each type of data is used for different purposes, first for training phase and second for testing phase.

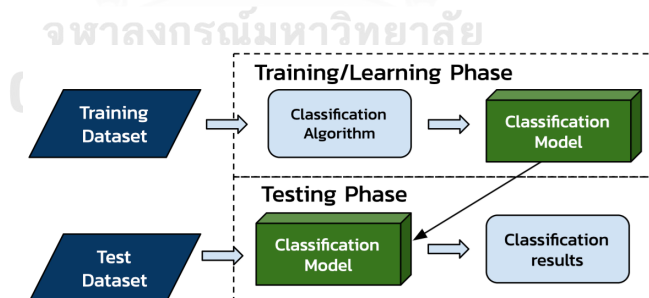


Figure 2.1: Classification process

Training/Learning phase

A training dataset, the set of labeled instances, is provided to the classification algorithm to train the classifier. Classification algorithms analyze the training data and use statistical methods and machine learning methods to discover hidden relationships between different features and the outcome that will be captured by model parameters. The output of the classification algorithm is a model called a classifier (a classification model).

Testing phase

The classification model predicts test instances for their class labels. The outcomes will be determined based on an analysis of the available features. They are in the form of values assigned to one or more categorical labels. Finally, the output is called classification results.

The performance of a classification model can be assessed by measuring the difference between the predicted class (classification results) with the actual class (actual results). Nevertheless, because the model is generally overfitted with the training dataset, it tends to accurately predict most instances in that dataset while misclassifying unseen instances from the testing dataset. As a result, the set of instances used in the training and testing processes should not be the same. The hold-out validation and the K-fold cross-validation are two extensively used methods for splitting a dataset [41] and evaluating the performance of a classifier.

- **Hold-out validation**

Generally, the hold-out validation divides data into two non-overlapping parts and these two parts are used for the training and testing datasets, respectively. The testing part is the “hold-out” part. Its name comes from we hold-out this part for testing and learn the model using the remainder part of the data. So, the hold-out validation can have different percentages

of the data being held out for testing, and note that the time using hold-out validation is relatively lesser than the time taken for learning by using k-fold cross validation. However, the hold-out validation sometimes divides data into three parts where the third part is the validation dataset to use for parameter tuning that is commonly applied before evaluating the model with the testing dataset.

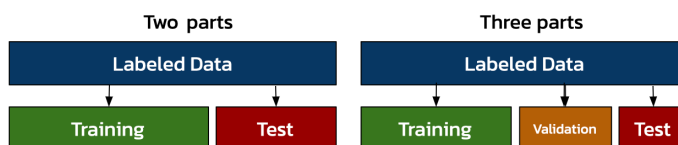


Figure 2.2: Dividing labeled data into two and three parts of hold-out validation

- **K-fold cross-validation**

The data is equally partitioned into k equal partitioned folds as in Figure 2.3. Training and testing on these partitioned folds are done in k iterations, with each iteration leaving one fold for testing and training the model on the remaining $k - 1$ folds. The model accuracy is calculated by averaging the accuracy obtained in each iteration.

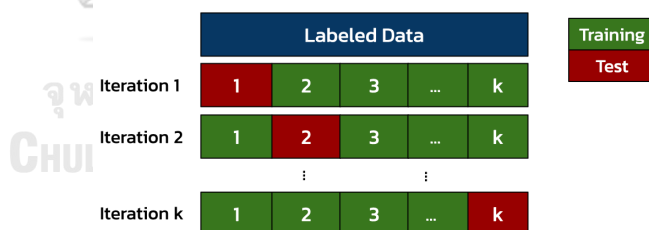


Figure 2.3: K-fold cross-validation

2.2 Training Dataset Used For Classification

The structure of the data set used in the research is presented in Figure 2.4. The number of instances, attributes, and classes in a training dataset is denoted by m , n , and k which are indexed by i and j , respectively. Let $A = \{A_j | j = 1, 2, \dots, n\}$ be a set of attributes and A_j represent the selected attribute. Define dataset $D = \{(x_i, y_i) | i = 1, 2, \dots, m\}$, and $D = D_1 \cup D_2 \cup \dots \cup D_k$ where $D_t = \{(x_i, y_i) \in D | y_i = c_t\}$. When $k = 2$, a training dataset is called a binary-class dataset while a training dataset has more than 2 classes ($k > 2$) will called a multi-class dataset.

		Attributes (a)				Class
		A_1	A_2	...	A_n	
Instances	x_1	$X_{1,1}$	$X_{1,2}$...	$X_{1,n}$	y_1
	x_2	$X_{2,1}$	$X_{2,2}$...	$X_{2,n}$	y_2
				...		
	x_m	$X_{m,1}$	$X_{m,2}$...	$X_{m,n}$	y_m

Figure 2.4 A dataset structure

2.3 Decision tree

A decision tree, one of the classification models used in machine learning, consists of root node, internal nodes, and leaf nodes like a normal tree structure. **The root node** having no incoming edges and one or more outgoing edges. **Internal nodes**, each of which has exactly one incoming edge and two or more outgoing edges and **leaf nodes or terminal nodes**, each of which has exactly one incoming edge and no outgoing edge. An attribute is on every internal node, the outcome is on branch and the class label, as a result, is on a leaf node [42][43].

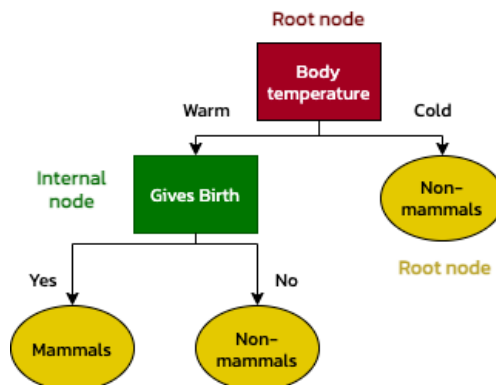
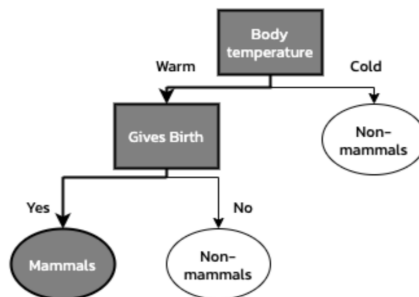


Figure 2.5: Example of a decision tree on the mammal classification problem [44]

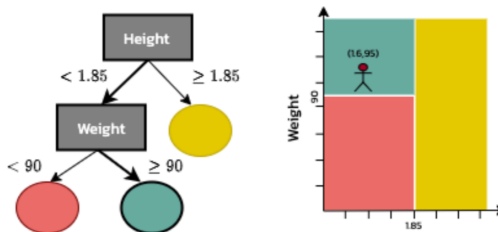
For example, as Figure 2.5 shows the decision tree for the mammal classification where the tree consists of a root node having no incoming edges and one or more outgoing edges, internal nodes, each of which has exactly one incoming edge and two or more outgoing edges and leaves or terminal nodes, each of which has exactly one incoming edge and no outgoing edge. The root node shown in Figure 2.5 uses the attribute “Body Temperature” to separate warm-blooded from cold-blooded vertebrates. Since all cold-blooded vertebrates are non-mammals, a leaf labeled “Non-mammals” is marked as the right child of the root node. If the vertebrate is warm-blooded, a subsequent attribute, “Gives Birth”, is used to distinguish mammals from other warm-blooded creatures, which are mostly birds.

Finally, classifying an instance from a test dataset is straightforward once a decision tree has been constructed. Starting from the root node, an instance is checked with the test condition at the root node and moves to the appropriate branch based on the outcome of the test. This will lead either to another internal node, for which a new test condition is applied or to a leaf where the class label associated with it is then assigned to an instance.

Name	Body Temperature	Gives Borth	...	Class
A	Warm	Yes	...	?



(a)



(b)

Figure 2.6: (a) predicts the class lebeled example, and (b) the case that all input attributes are continuous values

Two unlabeled instance classification examples are shown in Figure 2.6(a) and Figure 2.6(b). The decision tree evaluates this instance at the root node and moves through the internal nodes until the leaf node is encountered. The first example, predicts the class label of an animal named “A”, in which each leaf node, represents the class, either “A” is (“Mammals”) or is not (“Non-Mammals”). See Figure 2.6(a), the table above shows the training data animal named “A” having a warm body temperature and gives birth, the decision tree path terminates at leaf node labeled “Mammals”. In the case that all input attributes are continuous values, it gives the result as dividing the input space, as shown in the second example. The space R^2 is divided into three parts for each class, i.e.people who

are tall less than 1.85 m. and weighing less than 90 kg. class (red color), people who are tall less than 1.85 m., and weighing more than or equal to 90 kg. class (green color), and people taller or equal to 1.85 m. class (yellow color). The graph shows the unlabeled instance which is represented by the human head locating at (1.6, 95) is classified to be the second class.

2.3.1 Decision Tree Induction Algorithm

The algorithm works by recursively selecting to find the best attribute for splitting the data and expanding the leaf nodes of the tree until the stopping criterion is met. Commonly, a single attribute is used to select the criteria for splitting a set of instances in each non-leaf node. All possible scenarios for partitioning the dataset called candidates are discovered for each attribute, which varies depending on the type of attribute. There are numerical attribute and categorical attribute.

- **Categorical / Discrete attribute**

The data is divided into subsets based on the number of possible values of attributes. Figure 2.7(a), body temperature is the categorical attribute of training dataset D having two distinct values, warm-blooded and cold-blooded. Hence, the training data is divided into two subsets and in Figure 2.7(b), the data is divided into three subsets based on income categorical attributes Low, Medium, and High.

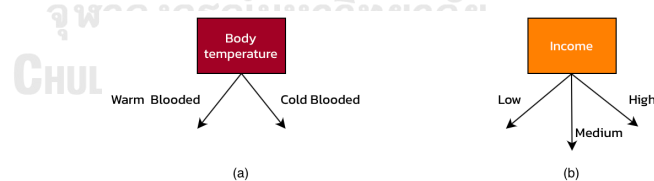


Figure 2.7: Example for partitioning the set of instances from the categorical attributes

• **Numerical / Continuous attribute**

Numerical attribute or continuous attribute has infinite possible values, e.g. a price of the house, annual company income, and etc. The numerical attribute could not divide into all values as the outcomes of the splitting condition. Thus, it frequently uses a particular value called the splitting values to be “split point” to partition the data into two subsets which depend on attribute A_j as Figure 2.8(a). Moreover, Figure 2.8(b) shows the income attribute having 20,000 to be split point for dividing data into two subsets. The first data is branched for the value less than 20,000 and the second is greater than or equal to 20,000. The condition to choose splitting value is computed from “the middle between each pair adjacent sequential value”, i.e. $\frac{X_{i,j}+X_{i+1,j}}{2}$ for $i = 1, 2, \dots, m - 1$, are all considered. The value that offers the “best splitting” measure is chosen to be the splitting value of attribute A_j .

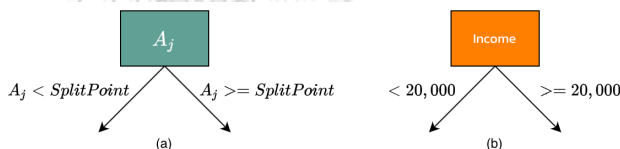


Figure 2.8: Example for partitioning the set of instances from the numerical attributes

See the details of the decision tree induction algorithm in Algorithm 2.1 generating decision tree, the inputs are training dataset D and set of attributes A . It begins with creating the root node for the entire training dataset. If all instances come from the same class, the leaf node is created corresponding to that class. Then, line 4 is the processing for finding the best attribute to be used for splitting in the tree, the detail of measure for selecting the best attribute demonstrated in section 2.3.2 after the algorithm. All possible values in the best attribute (A)

are used to branch the root below. Next, if the values within an internal node (Example(v_i)) are not empty, the algorithm is recursively called itself until remains only the root node.

Algorithm 1 GenerateDecisionTree (\mathbf{D} , attributes)

```

1: Create a root node for the tree
2: If all instances are in the same class then return the node labeled as that class
3: Otherwise Begin
4:  $A \leftarrow$  The attribute that best classifies examples(Attribute Selection Method)
5: Decision Tree attribute for Root =  $A$ 
6: for each possible value,  $v_i$ , of  $A$  do
7:   Add a new tree branch below Root, corresponding to the test  $A = v_i$ 
8:   Let Examples( $v_i$ ) be the subset of examples that have the value  $v_i$  for  $A$ 
9:   if Examples( $v_i$ ) is empty then
10:    below this new branch add a leaf node with a label (most target value)
11:   else
12:    below this new branch add the subtree
13:    GenerateDecisionTree (Examples( $v_i$ ), attributes –  $A$ )
14: End
15: Return root

```

2.3.2 Attribute Selection Measures

There are many measures for selecting “the best split” at each non-leaf node. Mostly, they are presented under the approach of measuring the impurity of dataset D , such as Shannon Entropy [45] and Gini index which are used in the classification like a regression tree (CART) [46]. Moreover, there are classification error, minority entropy, and minority condensation entropy which come to measure the criteria to split. Let $P_t(D)$ stand for the proportion of instances from class D_t , i.e. $P_t(D) = \frac{|D_t|}{|D|}$. The highest value of the measurement means the dataset contains a similar number of instances from all classes, and the lowest value is when all instances belong to the same class. See Figure 2.9, the comparison among the impurity measures, Entropy, Gini index, and the misclassification

error.

$$Entropy(\mathbf{D}) = - \sum_{t=1}^k P_t(\mathbf{D}) \log_2 P_t(\mathbf{D}) \quad (2.1)$$

$$Gini(\mathbf{D}) = 1 - \sum_{t=1}^k (P_t(\mathbf{D}))^2 \quad (2.2)$$

$$CMError(\mathbf{D}) = 1 - \max_{t=1,2,\dots,k} P_t(\mathbf{D}) \quad (2.3)$$

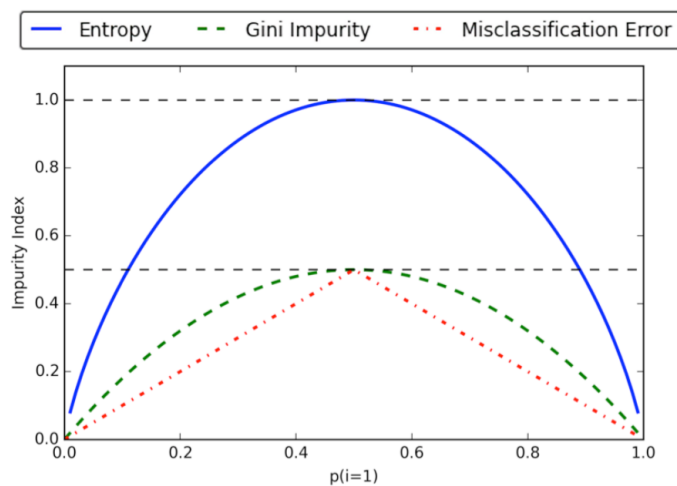


Figure 2.9: Comparison among the impurity measures for binary class-class dataset.

In 1986, the decision tree algorithm called the Iterative Dichotomiser 3 (ID3) algorithm [9] that applies **the information gain** (Equation (2.4)) to determine the splitting condition at each non-leaf node. It is the subtraction between the entropy of the entire dataset before splitting and the weighted average of the entropy of each partition after splitting, in which a candidate of the splitting condition a represents the partition of dataset \mathbf{D} into r subsets. The candidate providing the highest information gain is chosen to be the splitting condition. Nonetheless, using the information gain tends to favor the attribute having many distinct values. To remedy this situation, the decision algorithm known as the

C4.5 algorithm is introduced. It uses **the gain ratio**, shown in Equation (2.6), as the splitting measure that normalizes the information gain by dividing it with the split information defined by (2.5). Let $P^{(l)}(\mathbf{D}) = \frac{|\mathbf{D}^{(l)}|}{|\mathbf{D}|}$ where $P^{(l)}$ demonstrates the proportion of instances in $D^{(l)}$ compared to the entire dataset \mathbf{D} .

$$InfoGain_c(\mathbf{D}) = Entropy(\mathbf{D}) - Entropy_c(\mathbf{D}) \quad (2.4)$$

$$SplitInfo_c(\mathbf{D}) = - \sum_{l=1}^r P^{(l)}(\mathbf{D}) \log_2 P^{(l)}(\mathbf{D}) \quad (2.5)$$

$$GainRatio = \frac{InfoGain_c(\mathbf{D})}{SplitInfo_c(\mathbf{D})} \quad (2.6)$$

2.4 Random forests

In 2001, Breiman constructed a unified algorithm, supervised machine learning algorithm, called random forests [47]. The decision tree forms the base classifier in a random forest. This classifier combines the predictions made by multiple decision trees. As the named randomization is done in two ways in constructing random forests. One is bootstrapping for drawing subsamples (X_i) and the second is randomly selecting attributes (A_j) or features for generating decision trees. A general random forest algorithm for a tree-based model can be implemented as shown in Algorithm 2.3 [48].

2.4.1 Bootstrap Aggregation (Bagging)

Bootstrap Aggregation or Bagging [49] is a simple and all-powerful ensemble method, in which an ensemble method is a technique that combines the predictions from multiple machine learning algorithms together to make more accurate predictions than any individual model. Bagging can be used to decrease the vari-

ance for those algorithms that have high variance and is the application of the bootstrap procedure. Decision trees are highly dependent on the data they are trained on. If any of the training data is changed., e.g. a tree is trained on sub-samples of the training data, the resulting decision tree can be different and the predictions can be absolutely different.

Bootstrap (Efron, 1983) is an alternative to cross-validation that relies on random subsampling with replacement (see Figure 2.12 for an example). It is based on the assumption that samples are independently and identically distributed and it is especially recommended when the sample size is small [50]. After n extractions, the probability that a given object has not been extracted yet is equal to $(1 - \frac{1}{n})^n$. When n is sufficiently large, the probability asymptotically approaches $1 - e^{-1} \approx 0.632$. There are several variations to the bootstrap sampling approach in terms of how the overall accuracy of the classifier is computed. One of the more widely used approaches is the .632 bootstrap, The .632 bootstrap estimate of accuracy, introduced in Efron and Tibshirani (1997), is defined as:

$$Acc_{boot} = \frac{1}{b} \sum_{i=1}^b (0.632 acc_i + 0.368 acc_{train}) \quad (2.7)$$

where b is the number of generating a bootstrap sample, acc_i is the accuracy obtained with a model train on simple i and tested on the remaining instances, and acc_{train} is the accuracy of the model trained on the full training set.

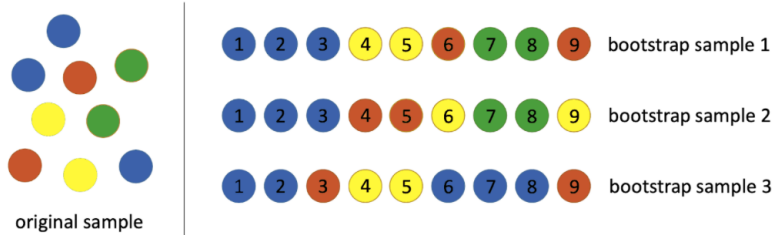


Figure 2.12: An example of bootstrap sampling.

Since objects are subsampled with replacement, some classes might be over-represented blue marbles in bootstrap samples 3 or even missing, green marbles in bootstrap sample 3.

2.4.2 Random Forest Model

See Figure 2.13 for the random forest model, the bagging approach is used in the steps in constructing the decision tree in the forest. Several decision tree classifiers are trained on bootstrap samples of the training data \mathbf{D} then let m be the number of attributes in the input dataset, and m_i represents the number of attributes to choose at each tree node where choose m attributes at random. The optimal split is computed based on the m_i input attributes of the subsampled dataset. Then, each tree is allowed to grow without being pruned. Finally, predictions on test data are obtained by combining the predictions of the trained classifiers with a majority voting scheme.

Algorithm 2 BasicRandomForests

```

1: input: Select the number of models to build,  $n$ 
2: for  $iteration = 1, 2, \dots, n$  do
3:   Generate a bootstrap sample of the training data
4:   Train a tree on this sample
5:   for each split do
6:     Randomly select  $k(\leq P)$  of the trainer predictors
7:     Select the best predictor among the  $k$  predictions
8:     and partition the data
9:   end
10:  Use typical tree model stopping criteria to determine
11:  when a tree is complete
12: end

```

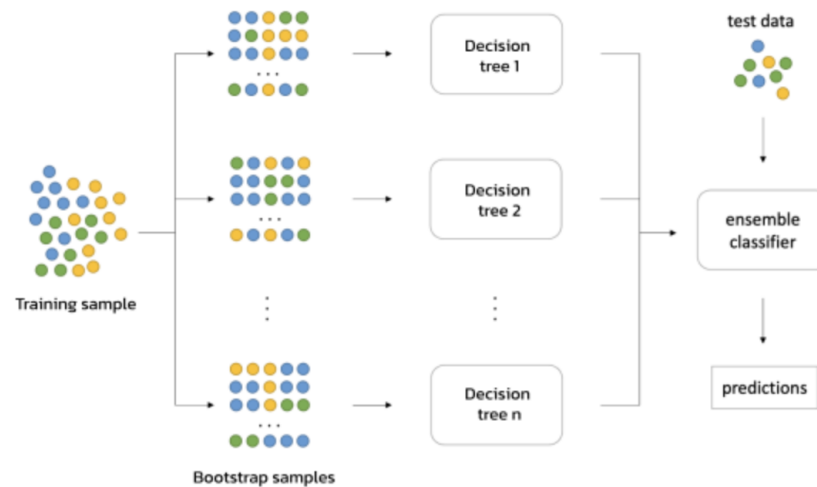


Figure 2.13: Random forest classification model

2.5 Class imbalanced Problem

Despite the fact that the problem of class imbalance has received a lot of attention in recent years, there is no precise way to define an imbalanced dataset. Technically, a dataset having unequal class distribution should be imbalanced. However, it is considered imbalanced when the number of instances in each class is significantly different [24].

2.5.1 Class Imbalance Classification Challenging

Class imbalance happens when there are significantly lesser training examples in one class compared to other classes. The nature of class imbalance distribution could occur in two situations [51]. First, when class imbalance is an intrinsic problem or it happens naturally. A naturally imbalanced class distribution happens in the case of credit card fraud or in rare disease detection. Second, when the data is not naturally imbalanced, instead it is too expensive to acquire such data for minority class learning due to cost, confidentiality, and tremendous effort to find

a well-represented data set, like a very rare occurrence of the failure of a space shuttle. Class imbalance involves a number of difficulties in learning, including imbalanced class distribution, training sample size, class overlapping, and small disjuncts. All these factors are explained in detail in the following sections.

2.5.2 Binary-Class Imbalanced Problem

Traditionally, the class imbalanced problem is often related to a binary-class dataset that one class containing significantly more instances than another class, called the binary-class imbalanced dataset. The majority class is normally represented by the negative class, while the minority class, will be focused for a classifier indicated by the positive class shown in Figure 2.17.

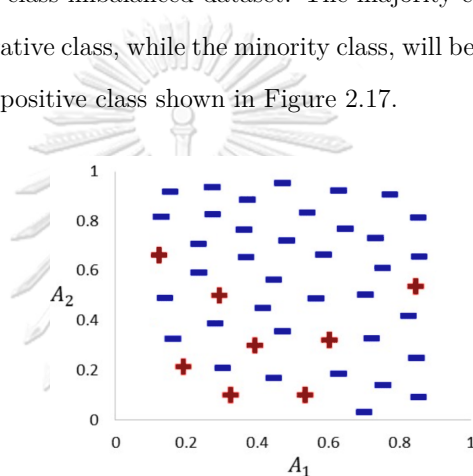


Figure 2.9: Binary-class imbalanced dataset

Mathematically, binary-class dataset \mathbf{D} containing m_- instances from the majority class, and m_+ instances from the minority class is said to be imbalanced when m_- is much greater than m_+ . To measure the degree of imbalanced for each binary-class dataset, the imbalanced ratio ($I.R.$) is used, which is defined by the ratio between the number of instances in the majority class and the minority class. For the problem of building the model to classify the binary-class imbalanced dataset, it is called the binary-class imbalanced problem. It is intended to predict the minority instances, while maintaining the correct classification of the majority

instances. Define as equation below:

$$I.R. = \frac{m_-}{m_+} \quad (2.8)$$

A binary-class imbalanced problem consists of four approaches[24]: there are (1) a data-level approach, (2) an algorithmic-level approach, (3) a cost-sensitive approach, and (4) an ensemble approach. The research is interested in using an algorithm-level approach for handling the problems by extending the idea of successful entropy that we will demonstrate in the section below and the research methodology will be described in the next chapter.

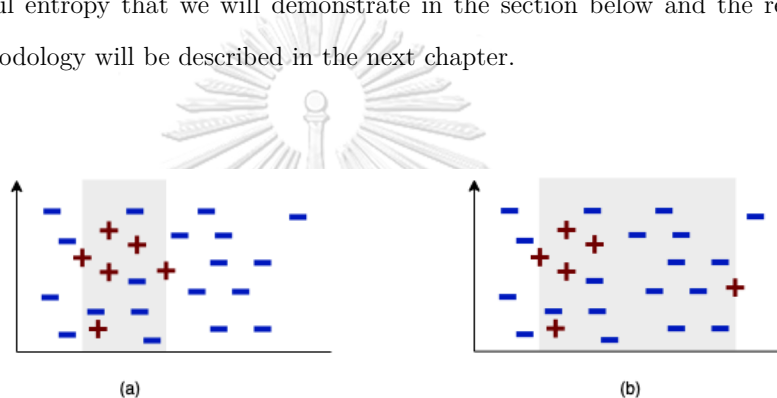


Figure 2.10: A difference of the minority range with the imbalanced data.

In 2019, an interesting splitting measure Minority Condensation Entropy (MCE), the entropy that is modified from minority entropy (ME), is also used to find the best attribute for constructing a decision tree and is designed to handle a binary-class imbalanced dataset. When the algorithm is faced with the biased class of the dataset, ME diminishing majority instances outside the range for all minority instances, called the minority range. See in Figure 2.10(a), the minority range is the size of the middle box, which provides sufficient information to construct a decision tree as illustrated at the top of the figure. All instances outside the range that would not affect the ability to predict minority class instances are excluded.

Hence, ME considers only instances in the minority range to construct a decision tree. Sometimes, if the imbalance dataset has the outlier as Figure 2.10(b), the minority range covers majority instances.

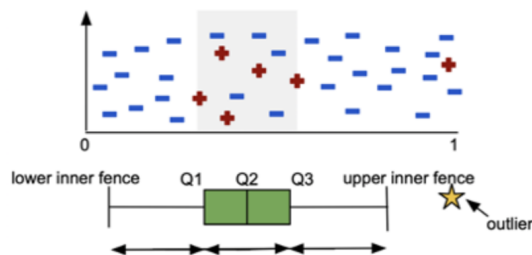


Figure 2.11: Applying the IQR rule to detect outliers before determining the minority instances.

The computation of MCE is based on the interquartile range (IQR) rule that is employed to the set of minority instance values for detecting the outliers. It defines the boundary that represents the range of acceptable values for the minority instances based on Tukey's boxplot [52]. The lower inner fence is defined by the first quartile minus 1.5 times of IQR, while the upper inner fence is defined by the third quartile plus 1.5 times of IQR. For example, Figure 2.11 demonstrates the use of the IQR rule. The set of instances within that range is considered, in which the minority class is more condensed. Let f be a function that maps each instance x_i in dataset \mathbf{D} to set of real numbers. For example with a two-dimensional, define function as $f(x_i) = f(x_{1,i}, x_{2,i}) = x_{1,i} + x_{2,i}$. Then, set of instance values $f(D) = \{f(x_i) \in R \mid (x_i, f(x_i)) \in \mathbf{D} \text{ for } i = 1, 2, \dots, m\}$ corresponds to \mathbf{D} , define $F(D)$ be the subset of $f(\mathbf{D})$ that ignores the outliers shown in (2.7) as equation below:

$$F(D) = \{f(x_i) \in f(\mathbf{D}) \mid Q1 - 1.5 * IQR \leq f(x_i) \leq Q3 + 1.5 * IQR\} \quad (2.9)$$

where $Q1$ is the first quartile of $f(\mathbf{D})$, $Q3$ is the third quartile of $f(\mathbf{D})$, and IQR is the interquartile range of $f(\mathbf{D})$ which is equal to $Q3 - Q1$. Thus, the minority range that ignores the outliers of $f(\mathbf{D})$ is determined by the range between the minimum value and the maximum value of $F(\mathbf{D}_+)$, i.e. $M_+(f(\mathbf{D})) = [\min F(\mathbf{D}_+), \max F(\mathbf{D}_+)]$. Then, $M_+(f(\mathbf{D}))$ implies the subset of instances within the minority range that ignores the outliers. Thus, the definition of the minority condensation entropy according to $f(\mathbf{D})$ is determined by (3.2) as follows:

$$MCE_{f(D)} = Entropy(M_+(f(D))) \quad (2.10)$$

Algorithm 3 $MCE(\mathbf{D}, f)$

- 1: **Input:** dataset \mathbf{D} , a function to obtain the instance values f
 - 2: **Output:** the minority condensation entropy of \mathbf{D} with respect to f
 - 3: Generate the set of instance values corresponding to the minority class without outliers: $F(\mathbf{D}_+)$
 - 4: Create the minority range that ignores the outliers: $M_+(f(\mathbf{D}))$
 - 5: Compute the subset of instances within the minority range that ignores the outliers: $M_+(f(\mathbf{D}))$
 - 6: **Return** $Entropy(M_+(f(\mathbf{D})))$
-

Pseudocode to compute MCE of dataset \mathbf{D} with respect to a function to obtain the set of instance values f is displayed in Algorithm 3. The minority range that ignores the outliers is generated to limit the set of instances before calculating the entropy. Finally, the decision tree built based on MCE is called Minority Condensation Decision Tree (MCDDT). An extension of the MCE to the random forest and the experiment of the enhanced algorithm will be detailed in the next chapter.

CHAPTER III

RESEARCH METHODOLOGY

In this chapter, an enhanced random forest for the classification of imbalanced data is proposed. It is called Random Forest of mixed Minority condensation Decision Tree or RMDT. The motivation of the proposed algorithm is presented along with the methodology and its algorithm. The last section demonstrates an example of a training tree in the proposed random forest classification which it describes the process and shows the structure of a tree having two different trees. First is the standard decision tree and second is the decision tree that builds based on the minority condensation entropy.

3.1 Random forest of mixed Minority condensation Decision Tree

Building the decision tree classifier based on MCE to handle the class imbalanced problem, fixed the problem of ME that sometimes it unnecessarily widens the minority range, which covers more majority instances because of having minority instance values extremely deviate from others within datasets. These reasons make MCDT highly successful in handling the class imbalanced problem and this performance can be improved with the ensemble learning method, in which multiple decision trees are combined as a random forest classifier.

Nonetheless, in the bootstrap of the random forest, when the algorithm is faced with the imbalanced distribution of the data, it has a chance to make minority instances disappear. For example, Figure 3.1(a) shows imbalanced dataset before bootstrapped and Figure 3.1(b) shows the dataset after bootstrapped. The prediction performances of the classifier are decreasing. As a result, the idea of keeping all minorities for bootstrapping comes from these reasons. However, the standard decision tree still outperforms for prediction when classifying a balanced

dataset. Therefore, the forest that has a mix of MCTD and standard decision trees should be used to develop for minority instances classification.

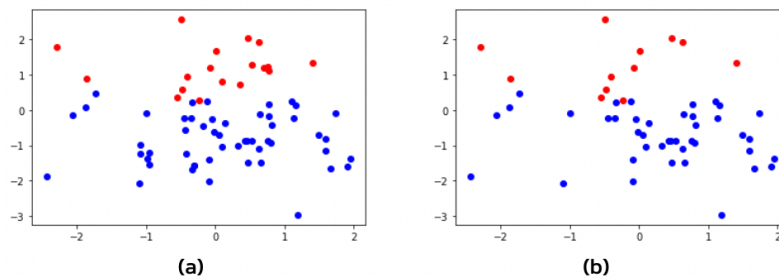


Figure 3.1: Imbalanced dataset before (a) and after bootstrapped (b)

3.2 Proposed Algorithm

As we mentioned above, this research proposes a random forest that uses a mixed standard decision tree and MCDT, and then we will call this enhanced random forest is RMDT, which has 2 construction parts in the training phase as follow:

1. It will bootstrap only majority instances and keep all minority instances in a training dataset's bootstrapping phase, ignoring the balance of these two classes, therefore subsamples from this part are imbalanced datasets, and MCE should be used to discover the best attribute to split in each decision tree. Visualization representation of the part is explained in Figure 3.1, red number 1 and 2 show the framework of first and second parts, respectively.

2. Bootstrapping also maintains all minority instances and bootstraps just majority instances, however in this part, we bootstrap until the number of majority instances and minority instances are equal, resulting in a dataset with balanced subsamples. The decision tree will then be built using SE.

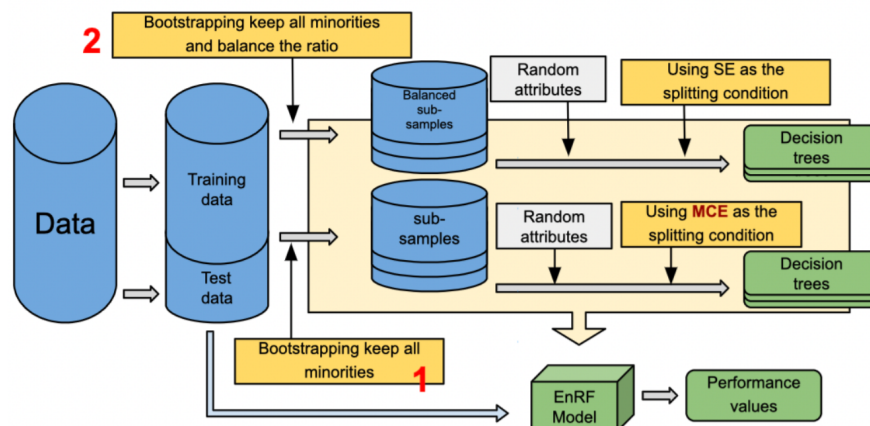


Figure 3.2: Framework of RMDT algorithm

The algorithm construction is described in Algorithm 4. The following parameters are required by the algorithm: N (number of standard trees trained in the forest) and n (number of minority condensation decision trees trained in the forest). Subsample computing is performed in Line 3 of Algorithm 4 for use in Line 5 and Line 7 which are balanced subsamples, and imbalanced subsamples, respectively.

Algorithm 4 RMDT

```

1: input: Learning data  $\mathbf{D}$ ,  $N$  and  $n$ 
2: for  $t = 1, 2, \dots, N + n$  do
3:   Computing balanced/imbalanced subsamples for each tree
4:   if  $t \leq n$  then
5:     DT = Built forest based on subsamples for standard decision trees
6:   else
7:     MCDT = Built forest based on subsamples for MCDT
8:   end
9:   RMDT = Combine two forests together
10: end of learning phase

```

For example, we synthesize 500 instances, and 85% of all instances are minority instances. The data distribution is in the graph on top of Figure 3.3. The number of training data is set to 70% of all instances. Illustrating the distribution of

training data and testing data are on the left and right sides below of Figure 3.3, Red dot denotes the minority class and Blue dot refers to the majority class. We can notice that minorities also disappear from separating to training data.

Next, the Training dataset is used by the bagging method. In our algorithm, we will set the size of the subsample equal to 2 in 3 of the number of training data. Users can determine the number of decision trees and standard decision trees by inputting the parameters n and N in the algorithm. In this example, we use two decision trees and two MCDTs for showing the structure of the tree. First, we will show tree structure by the example of the decision tree that is already trained with the training data as in Figure 3.4, and tree structure by the example of the MCDT that already fits with the training data seen in Figure 3.5. We can see that the decision tree and MCDT begin with the same feature which is the first feature. The algorithm will fit the data until the algorithm is done with the 4 trees. Then, the trained algorithm is used to prediction into the next step.

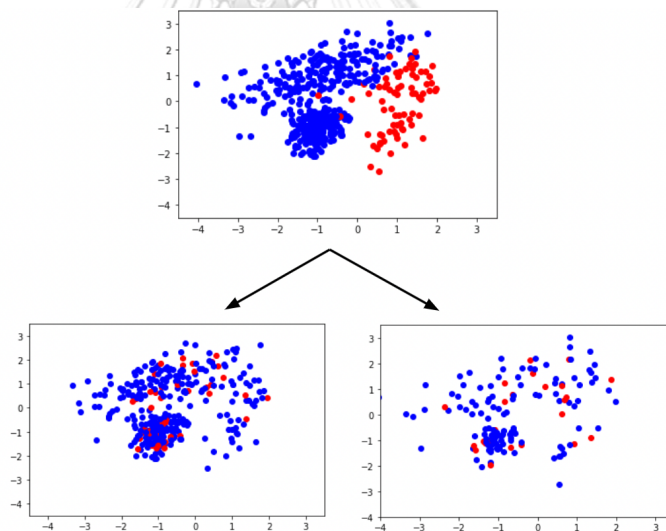


Figure 3.3: 2D imbalanced dataset, on top is the original dataset

Finally, our model has trained and then the trained model is used to predict

testing data and found the efficiency of the prediction which are the performance measures as Precision, Recall, and F-measure. In Chapter IV, computing the efficiency of RMDT, we will classify based on 2 datasets first from a synthetic dataset but more attributes or use in the high dimension and second by using the real-world datasets from UCI. Then, three performance measures are used for the classifier efficiency: precision, recall, F-measure, and G-measure. We will discuss the information on performance measures and each classifier model for comparison in the next chapter.

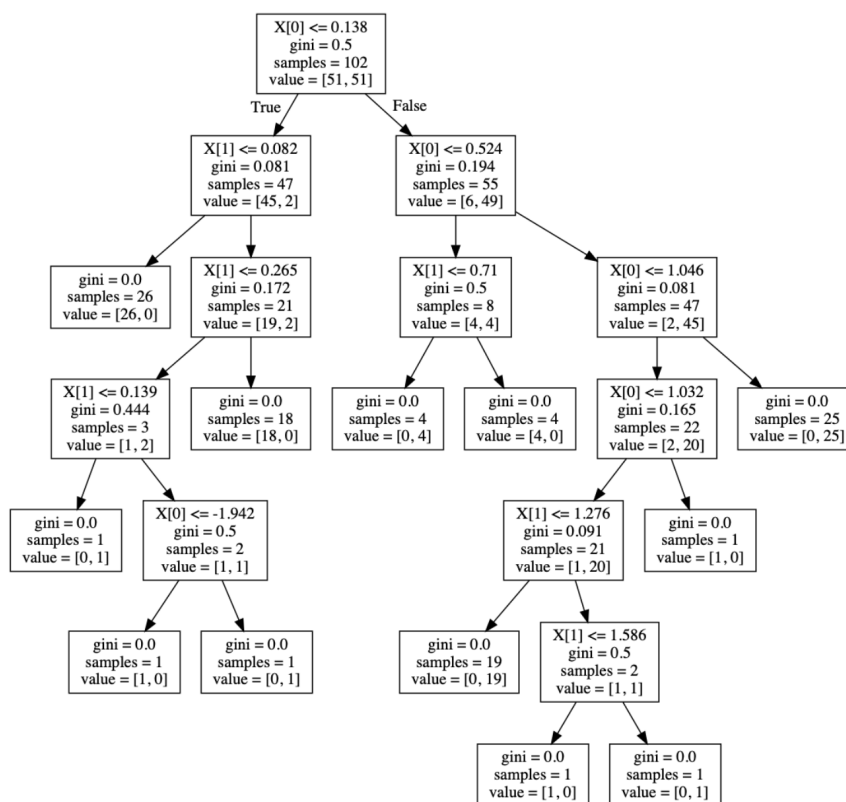


Figure 3.4: Example of a decision tree fitted structure

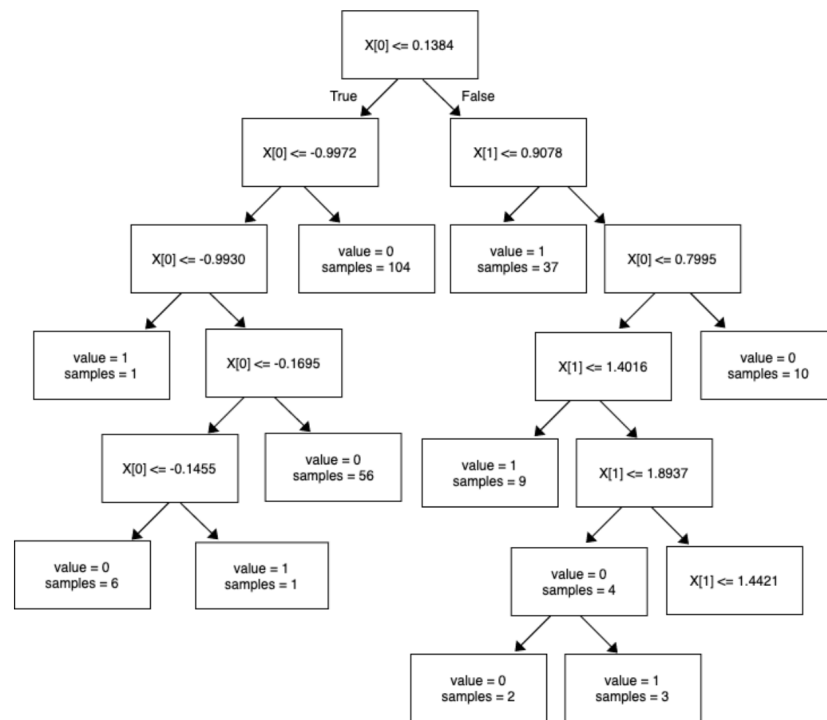


Figure 3.5: Example of the MCDT fitted structure



CHAPTER IV

EXPERIMENTS AND RESULTS

The RMCD algorithm is implemented via the Python programming language. All experiments in classifying on imbalanced data are presented in this section. The experiments are divided into two parts. The first part is an implementation for computation of the accuracy performance of RMDT compared with standard random forest based on a synthetic dataset having 500 instances. In the second part, the efficiency of the RMDT algorithm is compared with five algorithms, i.e standard random forest, MCDT, Random forest with class weighting, Random forest with bootstrap class weighting, and AdaBoost, based on ten real-world datasets.

4.1 Accuracy Performance

The datasets that will be used in this section are first described. The measures for evaluating each method's accuracy performance are then introduced. The selection of each parameter is then explained. Finally, the experiments and their results are shown, which is the most important part of this section.

Dataset

The experiments were performed on 20 imbalanced binary datasets. Ten of these datasets are from synthetic binary-class imbalanced numeric datasets and ten real-world application datasets from the UCI repository. Concisely, Table 4.1 summarizes the datasets utilized in this thesis.

Measurements

Four performance measures: Precision, Recall, F1-Measure, and Geometric mean are used for comparing the performance [53]. Those measures can be derived from the values in the confusion matrix (Table 4.1).

	Actual positive	Actual negative
Predicted positive	True positive (TP)	False positive (FP)
Predicted negative	False negative (FN)	True negative (TN)

Table 4.1: Confusion matrix

The entries in the confusion matrix are TP, FP, FN, and TN:

- **True Positive (TP)** is the number of positive instances that are correctly predicted as positive instances.
- **True Negative (TN)** is the number of negative instances that are correctly predicted as negative instances.
- **False Positive (FP)** is the number of negative instances that are incorrectly predicted as positive instances.
- **False Negative (FN)** is the number of positive instances that are incorrectly predicted as negative instances.

The following are four performance measures:

$$Precision = \frac{TP}{TP + FP} \quad (4.1)$$

$$Recall = \frac{TP}{TP + FN} \quad (4.2)$$

$$F - Measure = (\beta + 1) \times \frac{Precision \times Recall}{\beta \times (Precision + Recall)} \quad (4.3)$$

$$G - Measure = \sqrt{Precision \times Recall} \quad (4.4)$$

The precision is the percentage of the correctly predicted minority (positive) class from the model. For the recall, it shows the proportion of the number of minority classes that are detected from the model. For example, if almost all predicted minorities are correct, but there are many other minorities that are not detected, then the precision is high and the recall is low. On the other hand, if almost all minorities are detected, but there are many incorrectly predicted minorities, then the precision is low and the recall is high. The G-measure is the geometric mean of Precision and Recall. Finally, the F-Measure is the harmonic mean of precision and recall in which we used beta is 1 which means Precision and Recall are equally important. Then, the bigger F-Measure value has greater overall precision and recall than the smaller F-Measure.

	Actual positive	Actual negative
Predicted positive	$TP = 20$	$FP = 5$
Predicted negative	$FN = 10$	$TN = 50$

Table 4.2: Example of the confusion matrix.

For example, the results of classifying the binary-class dataset are presented by the confusion matrix in Table 4.1. Therefore, precision, recall, F-measure, and Geometric mean are computed by (4.1), (4.2), (4.3), and (4.4), respectively. For precision which equals 0.8, it means that the instances that are predicted as the positive class have an 80% chance that they are actually positive. For recall which equals 0.67, it means that the instances that are actually positive have a 67% chance that they are predicted as the positive class. Then, the harmonic mean between these two measures is represented by the F-measure which equals 0.73, the same as the geometric mean value.

$$Precision = \frac{20}{20 + 5} = 0.80 \quad (4.5)$$

$$Recall = \frac{20}{20 + 10} = 0.67 \quad (4.6)$$

$$F - Measure = 2 \times \frac{0.8 \times 0.67}{0.8 + 0.67} = 0.73 \quad (4.7)$$

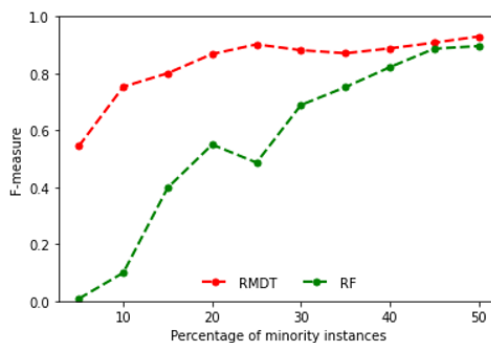
$$G - Measure = \sqrt{0.8 * 0.67} = 0.73 \quad (4.8)$$

4.2 Experiment Results and discussion

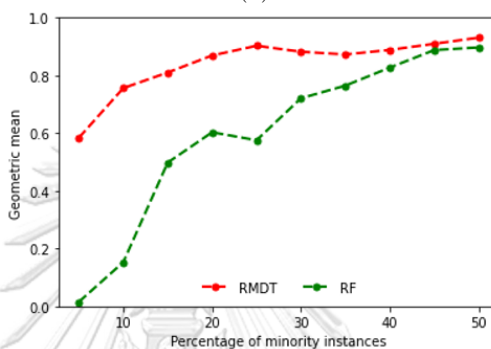
An enhancement of the standard random forest (RF) to classify minority instances in the binary-class imbalanced datasets dealing with numeric attributes using SE and MCE are exhibited in the experiments on collections of synthetic dataset according to Section 4.2.1. Accordingly, the average results of RF and RMDT are compared via the F-measure (4.3) and the G-measure (4.4) with respect to the minority class and the majority class displayed in Figures 4(a) and 4(b), respectively.

4.2.1 Synthetic dataset

The synthetic datasets used in this section are generated by Sklearn's function [54] which is a synthetic binary-class imbalance numeric dataset consisting of 500 instances having 50 attributes. There are ten groups of experiments having different percentages of the minority from 5% to 50%. The description of the results is presented in Figure 4.1 below.



(a)



(b)

Figure 4.1: The experimental results on synthetic datasets varying percentage of minority instances comparing with Standard random forest (RF) via F-measure (a) and Geometric mean (b)

For the results, F-measure and Geometric mean values of both RMDT (red line) and RF (green line) increase when the percentage of minority instances increases. Evidently, RMDT significantly outperforms RF when the number of minority instances is tiny, while their values will approach 1 when a dataset is more balanced. It is because RF tends to focus on the class having a large number of instances, while RMDT tries to make them balanced before considering them. Therefore, these results confirm that RF is ineffective in dealing with binary-class imbalanced problems.

4.2.2 Real-world dataset

The real-world binary-class datasets which are used in this thesis are provided from the UCI repository [55]. In Table 4.3, they are sorted in descending order by the percentage of instances in the minority class (%Min.). The first two columns indicate the number and the name of each dataset. For the number of instances (#Inst.) and the number of attributes (#Att.), they are shown in the third column and the fourth column, respectively. Particularly, the minority class and the majority class are presented in the fifth column. In order to evaluate the performance of each method, the experiments are repeated 50 times. See Table 4.3 for their descriptions, including the number of instances, the number of features, the percentage of minorities, and the imbalanced ratio.

No.	Datasets	#Inst	#Att	Min/Maj	%Min	I.R.
1	<i>Pima</i>	768	8	'1'/'0'	34.9	1.87
2	<i>Stakotlog Vehicle</i>	846	18	'bus'/The rest	25.77	2.88
3	BeastTissue	106	9	'fad'/The rest	14.15	6.07
4	NewThyroid	215	5	'3'/The rest	13.95	6.17
5	Fertility	100	9	'O'/'N'	12	7.33
6	Ecoli	336	7	'imU'/The rest	10.42	8.6
7	OpticDigits	1108	641	'8'/The rest	9.86	9.14
8	Glass	214	9	'5'/The rest	6.07	15.46
9	Wine quality-red	1599	11	'3'/The rest	3.94	24.38
10	Yeast	1484	8	'VAC'/The rest	2.02	48.47

Table 4.3: The characteristics of real-world binary-class datasets used in the experiments.

Moreover, to demonstrate the effectiveness of RMDT on a general dataset, the random forest built based on MCE and SE is evaluated with experiments on real-world datasets. The results are compared to those of five other classifiers. The first is a standard random forest. Additionally, the decision tree built based on MCE called MCDT is used as well. Then, random forest with class weighting, and

random forest with bootstrap class weighting [56] are the algorithm that works by weighting the instances and is used for comparison also. Lastly, the popular boosting algorithm, like AdaBoost [57][58] is also used in the comparison. It works by first fitting a decision tree on the dataset, then determining the errors made by the tree and weighing the examples in the dataset by those errors so that more attention is paid to the misclassified examples and less to the correctly classified examples. A subsequent tree is then fit on the weighted dataset intended to correct the errors. The process is then repeated for a given number of decision trees.

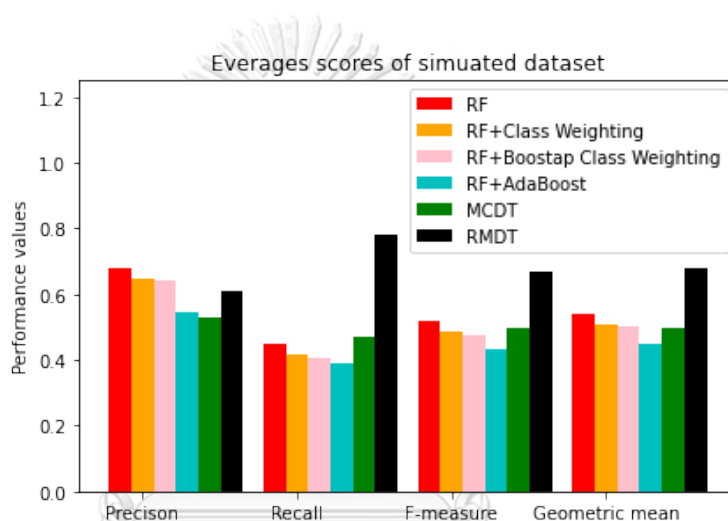


Figure 4.2: The experimental results on real-world datasets comparing by the average performance

In order to evaluate the dataset into the training set and the testing set, they are repeated 10 times. Accordingly, the average results of each classifier are compared via precision, recall, F-measure, and Geometric mean. Graphically, the bar chart representing the comparison of the average performance corresponding to each performance measure is shown in Figure 4, in which the higher value indicates the better performance, the black bar denotes RMDT performance val-

ues, the green bar denotes MCDT performances values, blue bar denotes Random Forest using Adaboost (RF+Adaboost)'s values, pink bar denotes Random Forest using bootstrap class weighting (RF+Bootstrap Class Weighting), orange bar and red bar denote Random forest using Class weight (RF+Class Weighting) and standard Random Forest (RF) performance values, respectively. Comparing the precision of all classifiers, RMDT yields similar performance at 0.611 to RF, RF+Weighting, and RF+Bootstrap Class Weighting but higher than RF+AdaBoost, and MCDT. It means that the number of RMDT predicted minority instances to be the majority class is lower than RF+AdaBoost and MCDT but similar to RF, RF+Weighting, and RF+Bootstrap Class Weighting.

Furthermore, for comparison by recall, RMDT yields the highest average performance at 0.778, which is much different from MCDT, RF, RF+Class Weighting, RF+Bootstrap Class Weighting, and RF+Adaboost. They yield the second-highest, the third-highest, the fourth-highest, the fifth-highest, and the sixth-highest average performance at 0.471, 0.446, 0.414, 0.408, and 0.389, respectively. This means that the number of they predicted majority instances to be the minority class has more than RMDT. For comparison by F-measure and Geometric mean, they are not exhibiting the different results. RMDT yields the highest average performance at 0.670 and 0.681, which is better than other classifiers, respectively.

CHAPTER V

CONCLUSION

This research proposed an enhanced random forest called RMDT which is a random forest that uses both the standard decision trees and the Minority Condensation Decision Tree that successfully handles the class imbalanced problem, which arises from extending the minority entropy concept. The improved performance to classify an imbalanced dataset of the research proposed, RMDT, is shown by two collections of experiments which are on synthetic datasets of binary-class imbalanced numerics and real-world binary-class datasets from UCI, respectively.

In the first experiment, the synthetic datasets are used for classification. The results show that RMDT outperforms the standard random forest when the number of minority instances decreases, and then their values approach the same values when the dataset is more balanced. These apparently confirm the standard random forest is not suitable for dealing with the binary-class imbalanced problem. Additionally, in the second experiment, RMDT performs the results better than standard random forest, Minority Condensation Decision Tree, Random forest with class weighting, Random forest having bootstrap class weighting, and Random forest mixed Adaboost on recall measure, F-measure, and Geometric mean. Especially recall, the enhanced random forest, RMDT shows the highest value which indicates that the number of the majority instances that are predicted to be the minority class has less than other comparing classifiers.

Future Works

Although RMDT is successful in handling the class imbalanced problem, there is considerable room for future work. The proposed algorithm still has to be extended in order to function on more complex datasets, such as multi-class imbal-

anced datasets, and multi-class with categorical attributes imbalanced datasets. Additionally, researching a new multi-criteria decision-making method that is intriguing to be applied in the decision tree is interesting for improving the construction of RMDT.



REFERENCES

- [1] Wikipedia contributors, “History of artificial intelligence — Wikipedia, the free encyclopedia,” 2022. [Online; accessed 6-October-2022].
- [2] . . . สงวนสิทธิ์, *Artificial Intelligence with Machine Learning, AI สร้างได้ด้วยแมชชีนเลิร์นนิ่ง*.
- [3] M. Block, M. Bader, E. Tapia, M. Ramírez, K. Gunnarsson, E. Cuevas, D. Zaldivar, and R. Rojas, “Using reinforcement learning in chess engines,” *Research in Computing Science*, pp. 31–40, 2008.
- [4] O. Theobald, *Machine Learning for Absolute Beginners: A Plain English Introduction*. Machine Learning from Scratch Series, Scatterplot Press., 2017.
- [5] A. Mujianto, C. Mashuri, A. Andriani, and F. Jayanti, “Consumer customs analysis using the association rule and apriori algorithm for determining sales strategies in retail central,” *E3S Web of Conferences*, vol. 125, p. 23003, 01 2019.
- [6] T. M. Cover and P. E. Hart, “Nearest neighbor pattern classification,” *IEEE Trans. Inf. Theory*, vol. 13, pp. 21–27, 1967.
- [7] H. ZHANG, “Exploring conditions for the optimality of naïve bayes,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, no. 02, pp. 183–198, 2005.
- [8] R. E. Wright, “Logistic regression.,” 1995.
- [9] J. R. Quinlan, “Induction of decision trees,” *Mach. Learn.*, vol. 1, p. 81–106, mar 1986.
- [10] J. Quinlan, “Simplifying decision trees,” *International Journal of Man-Machine Studies*, vol. 27, no. 3, pp. 221–234, 1987.
- [11] A. Cutler, D. R. Cutler, and J. R. Stevens, *Random Forests*, pp. 157–175. Boston, MA: Springer US, 2012.
- [12] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92*, (New York, NY, USA), p. 144–152, Association for Computing Machinery, 1992.
- [13] I. V. Tetko, D. J. Livingstone, and A. I. Luik, “Neural network studies. 1. comparison of overfitting and overtraining,” *Journal of Chemical Information and Computer Sciences*, vol. 35, no. 5, pp. 826–833, 1995.

- [14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.
- [15] B. Krawczyk, M. Galar, L. Jeleń, and F. Herrera, "Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy," *Appl. Soft Comput.*, vol. 38, p. 714–726, jan 2016.
- [16] S. Prasath, "Polyp detection and segmentation from video capsule endoscopy: A review," *Journal of Imaging*, vol. 3, 12 2016.
- [17] N. H. Vo and Y. Won, "Classification of unbalanced medical data with weighted regularized least squares," in *Proceedings of the 2007 Frontiers in the Convergence of Bioscience and Information Technologies*, FBIT '07, (USA), p. 347–352, IEEE Computer Society, 2007.
- [18] Y. Sahin, S. Bulkan, and E. Duman, "A cost-sensitive decision tree approach for fraud detection," *Expert Syst. Appl.*, vol. 40, p. 5916–5923, nov 2013.
- [19] W. Wei, J. Li, L. Cao, Y. Ou, and J. Chen, "Effective detection of sophisticated online banking fraud on extremely imbalanced data," *World Wide Web*, vol. 16, pp. 449–475, 2012.
- [20] E. Blanzieri and A. Bryl, "A survey of learning-based techniques of email spam filtering," *Artificial Intelligence Review*, vol. 29, 03 2008.
- [21] X. Ma, J. Guo, and X. Sun, "Dnabp: Identification of dna-binding proteins based on feature selection using a random forest and predicting binding residues," *PLOS ONE*, vol. 11, pp. 1–20, 12 2016.
- [22] S. Razakarivony and F. Jurie, "Vehicle detection in aerial imagery : A small target detection benchmark," *J. Vis. Commun. Image Represent.*, vol. 34, pp. 187–203, 2016.
- [23] T. Munkhdalai, O.-E. Namsrai, and K. Ryu, "Self-training in significance space of support vectors for imbalanced biomedical event data," *BMC bioinformatics*, vol. 16 Suppl 7, p. S6, 04 2015.
- [24] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, "Learning from imbalanced data sets," in *Springer International Publishing*, 2018.
- [25] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, jun 2002.

- [26] J. Maudes, J. Rodríguez, C. García-Osorio, and N. García-Pedrajas, “Random feature weights for decision tree ensemble construction,” *Information Fusion*, vol. 13, pp. 20–30, 01 2012.
- [27] L. Gonzalez-Abril, H. Nuñez, C. Angulo, and F. Velasco, “Gsvm: An svm for handling imbalanced accuracy between classes inbi-classification problems,” *Applied Soft Computing*, vol. 17, pp. 23–31, 2014.
- [28] D. A. Cieslak, T. R. Hoens, N. Chawla, and W. P. Kegelmeyer, “Hellinger distance decision trees are robust and skew-insensitive,” *Data Mining and Knowledge Discovery*, vol. 24, pp. 136–158, 2011.
- [29] A. Sagoolmuang and K. Sinapiromsaran, “Oblique decision tree algorithm with minority condensation for class imbalanced problem,” *Engineering Journal*, vol. 24, no. 1, pp. 221–237, 2020.
- [30] A. Cano, A. Zafra, and S. Ventura, “Weighted data gravitation classification for standard and imbalanced data,” *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1672–1687, 2013.
- [31] C. Diamantini and D. Potena, “Bayes vector quantizer for class-imbalance problem,” *IEEE Trans. Knowl. Data Eng.*, vol. 21, pp. 638–651, 05 2009.
- [32] M. Ohsaki, P. Wang, K. Matsuda, S. Katagiri, H. Watanabe, and A. Ralescu, “Confusion-matrix-based kernel logistic regression for imbalanced data classification,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 9, pp. 1806–1819, 2017.
- [33] Z.-H. Zhou and X.-Y. Liu, “Training cost-sensitive neural networks with methods addressing the class imbalance problem,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2006.
- [34] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.
- [35] L. Breiman, “Bagging predictors,” *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.
- [36] A. Sagoolmuang, “Self-balancing recursive partitioning algorithm for classification problems,” 2019.

- [37] K. Boonchuay, K. Sinapiromsaran, and C. Lursinsap, "Decision tree induction based on minority entropy for the class imbalance problem," *Pattern Anal. Appl.*, vol. 20, p. 769–782, aug 2017.
- [38] D. Punjani and K. Atkotiya, "A comprehensive study of various classification techniques in medical application using data mining," 07 2018.
- [39] J. Han, M. Kamber, and J. Pei, "8 - classification: Basic concepts," in *Data Mining (Third Edition)* (J. Han, M. Kamber, and J. Pei, eds.), The Morgan Kaufmann Series in Data Management Systems, pp. 327–391, Boston: Morgan Kaufmann, third edition ed., 2012.
- [40] C. Aggarwal, *Data Mining: The Textbook*. Springer International Publishing, 2015.
- [41] S. Yadav and S. Shukla, "Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification," pp. 78–83, 02 2016.
- [42] S. D. Jadhav and H. Channe, "Efficient recommendation system using decision tree classifier and collaborative filtering," 2016.
- [43] A. Gershman, A. Meisels, K.-H. Lüke, L. Rokach, A. Schclar, and A. Sturm, "A decision tree based recommender system," in *10th International Conference on Innovative Internet Community Systems (I2CS) – Jubilee Edition 2010* – (G. Eichler, P. Kropf, U. Lechner, P. Meesad, and H. Unger, eds.), (Bonn), pp. 170–179, Gesellschaft für Informatik e.V., 2010.
- [44] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Pearson International Edition, Pearson Addison Wesley, 2006.
- [45] R. Chen, "A brief introduction on shannon's information theory," 01 2016.
- [46] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. Routledge, 2017.
- [47] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [48] M. Kuhn, K. Johnson, *et al.*, *Applied predictive modeling*, vol. 26. Springer, 2013.
- [49] J. Brownlee, *Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch*. Machine Learning Mastery, 2016.
- [50] P. Galdi and R. Tagliaferri, *Data Mining: Accuracy and Error Measures for Classification and Prediction*. 01 2018.

- [51] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, "Classification with class imbalance problem," *Int. J. Advance Soft Compu. Appl*, vol. 5, no. 3, 2013.
- [52] Wikipedia contributors, "Box plot — Wikipedia, the free encyclopedia," 2022. [Online; accessed 7-October-2022].
- [53] M. K. Buckland and F. C. Gey, "The relationship between recall and precision," *J. Am. Soc. Inf. Sci.*, vol. 45, pp. 12–19, 1994.
- [54] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [55] C. B. D. Newman and C. Merz, "UCI repository of machine learning databases," 1998.
- [56] J. Brownlee, *Imbalanced Classification with Python: Better Metrics, Balance Skewed Classes, Cost-Sensitive Learning*. Machine Learning Mastery, 2020.
- [57] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.
- [58] P. Bahad and P. Saxena, "Study of adaboost and gradient boosting algorithms for predictive analytics," 2019.

BIOGRAPHY

Name	Miss. Suvaporn Homjandee
Date of Birth	February 12, 1998
Place of Birth	Ratchaburi, Thailand
Educations	B.Sc. (Mathematics) (First Class Honours), Kasetsart University, 2014
Scholarships	Development and Promotion of Science and Technology Talents Project (DPST)
Publications	

- S. Homjandee and K. Sinapiromsaran, A Random Forest with Minority Condensation and Decision Trees for Class Imbalanced Problems, *International Conference on Circuits Systems Communications and Computers*, vol. 16, pp. 502–507, 2021.