กรอบการทำงานสุ่มเพิ่มกลุ่มข้อมูลด้อยด้วยความหนาแน่นสำหรับปัญหากลุ่มข้อมูลอสมดุล

นายชุมพล บุญคุ้มพรภัทร

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

THE DENSITY-BASED MINORITY OVER-SAMPLING FRAMEWORK FOR CLASS IMBALANCED PROBLEMS

Mr. Chumphol Bunkhumpornpat

A Dissertation Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy

Program in Computer Science

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic year 2011

Thesis Title	THE DENSITY-BASED MINORITY OVER-SAMPLING FRAMEWORK FOR CLASS IMBALANCED PROBLEMS
By	Mr. Chumphol Bunkhumpornpat
Field of Study	Computer Science
Thesis Advisor	Assistant Professor Krung Sinapiromsaran, Ph.D.
Thesis Co-advisor	Professor Chidchanok Lursinsap, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

…….. Dean of the Faculty of Science
(Professor Supot Hannongbua, Dr.rer.nat.)

THESIS COMMITTEE

…….. Chairman
(Assistant Professor Jaruloj Chongstitvatana, Ph.D.)

…….. Thesis Advisor
(Assistant Professor Krung Sinapiromsaran, Ph.D.)

…….. Thesis Co-advisor
(Professor Chidchanok Lursinsap, Ph.D.)

…….. Examiner
(Assistant Professor Saranya Maneeroj, Ph.D.)

…….. Examiner
(Siripun Sanguansintukul, Ph.D.)

…….. External Examiner
(Kamol Keatruangkamala, Ph.D.)

ชุมพล บุญคุ้มพรภัทร : กรอบการทำงานสุ่มเพิ่มกลุ่มข้อมูลด้อยด้วยความหนาแน่นสำหรับปัญหากลุ่มข้อมูลอสมดุล. (THE DENSITY-BASED MINORITY OVER-SAMPLING FRAMEWORK FOR CLASS IMBALANCED PROBLEMS) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร. กรุง สินอภิรมย์สราญ, อ. ที่ปรึกษาวิทยานิพนธ์ร่วม : ศ.ดร. ชิดชนก เหลือสินทรัพย์, 70 หน้า.

เซตข้อมูลจัดอยู่ในปัญหากลุ่มข้อมูลอสมดุลเมื่อกลุ่มข้อมูลเป้าหมายมีจำนวนข้อมูลน้อยมากเปรียบเทียบกับกลุ่มข้อมูลอื่น ตัวจำแนกกลุ่มข้อมูลโดยทั่วไปมีความผิดพลาดในการทำนายกลุ่มข้อมูลด้อยนี้เพราะจำนวนข้อมูลในกลุ่มมีขนาดเล็ก วิทยานิพนธ์ฉบับนี้ได้นำเสนอกรอบการทำงานสุ่มเพิ่มกลุ่มข้อมูลด้อยด้วยความหนาแน่น กรอบการทำงานนี้ถูกออกแบบให้สุ่มเพิ่มข้อมูลในกลุ่มข้อมูลรูปร่างทั่วไป โดยใช้หลักความหนาแน่นของกลุ่มข้อมูล กล่าวโดยละเอียด กรอบการทำงานนี้สร้างข้อมูลสังเคราะห์ตามแนววิถีสั้นสุดระหว่างข้อมูลแต่ละตัวและจุดเซนทรอยด์เทียมในกลุ่มข้อมูลของกลุ่มข้อมูลด้อย ดังนั้น เซตของข้อมูลสังเคราะห์มีความหนาแน่นใกล้จุดเซนทรอยด์เทียมและมีความเบาบางไกลจุดเซนทรอยด์เทียม จากการกระจายของเซตข้อมูลดังกล่าว ตัวจำแนกกลุ่มข้อมูลเน้นการเรียนรู้บริเวณแกนมากกว่าบริเวณขอบของกลุ่มข้อมูล ผลการทดลองแสดงให้เห็นว่ากรอบการทำงานนี้พัฒนา ความแม่นยำ ค่าเอฟ (เทอมรวมของพรีซิชันและรีคอล) และ เอยูซี มากกว่าขั้นตอนวิธีสโมทและเซฟเลเวลสโมท

ภาควิชา<u>คณิตศาสตร์และวิทยาการคอมพิวเตอร์</u> ลายมือชื่อนิสิต.....

สาขาวิชา<u>วิทยาการคอมพิวเตอร์</u> ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก.....

ปีการศึกษา <u>2554</u> ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์ร่วม.....

5073820723 : MAJOR COMPUTER SCIENCE

KEYWORDS : CLASS IMBALANCED / OVER-SAMPLING / DENSITY-BASED

CHUMPHOL BUNKHUMPORNPAT : THE DENSITY-BASED MINORITY OVER-SAMPLING FRAMEWORK FOR CLASS IMBALANCED PROBLEMS.

ADVISOR : ASST. PROF. KRUNG SINAPIROMSARAN, Ph.D.

CO-ADVISOR : PROF. CHIDCHANOK LURSINSAP, Ph.D., 70 pp.

A dataset embodies the class imbalanced problem when the target class has a very small number of instances relative to the other classes. A trivial classifier typically fails to predict the positive instances due to its tiny size. In this thesis, the density-based minority over-sampling framework is proposed. It relies on a density-based notion of clusters and is designed to over-sample an arbitrarily shaped cluster discovered by the density-based clustering algorithm. In detail, my framework generates a synthetic instance along the shortest path from each instance in a cluster of a minority class to the pseudo-centroid of this cluster. Consequently, a set of the synthetic instances is dense near the pseudo-centroid and is sparse far from this centroid. Due to the distribution of the set, a classifier faces more emphatically around the core region than it does around the border region. The experimental results show that my framework improves accuracy, F-value (combination term of Precision and Recall), and AUC of a classifier more than SMOTE and Safe-Level-SMOTE.

Department : Mathematics and Computer Science Student's Signature .....

Field of Study : Computer Science Advisor's Signature .....

Academic Year : 2011 Co-advisor's Signature .....

Acknowledgements

I am heartily thankful to my advisor, Asst. Prof. Dr. Krung Sinapiromsaran, and my co-advisor, Prof. Dr. Chidchanok Lursinsap, whose encouragement, guidance and support from the initial to the final step enabled me to develop an understanding of the subject. I would like to thank my examiners, Asst. Prof. Dr. Jaruloj Chongstitvatana, Asst. Prof. Dr. Saranya Maneeroj, Dr. Siripun Sanguansintukul, and Dr. Kamol Keatruangkamala, whose gave me useful comments on my research work.

This research is supported by grant funds from the program Strategic Scholarships for Frontier Research Network for the Ph.D. Program Thai Doctoral degree from the Commission on Higher Education, Thailand.

Contents

Chapter Page

List of Tables

List of Figures

CHAPTER I

Introduction

In the first chapter, according to my thesis proposal, goal, scope, achievement, and research methodology are described as follows.

My thesis goal is to develop an integration technique of DBSCAN and DBSMOTE to over-sample a minority class in an imbalanced dataset; as a result, the classification performance of a classifier is improved.

My framework relies on a density-based concept to operate on an imbalanced dataset with multiple minority classes. After applying my density-based framework for handling the class imbalanced problem I expect to achieve the significant improvement of decision tree C4.5, RIPPER, and multilayer perceptron (MLP), when evaluating on accuracy, F-value, and AUC.

I design a new data structure as a connected graph for the over-sampling purpose. In my framework, I construct the graph from a cluster of a minority class and then generate a synthetic instance along the path between each instance and the pseudo-centroid of this cluster. Consequently, the synthetic instances are dense nearby the centroid and sparse far from this centroid. The distribution of the synthetic instances prevents the overlapping problem and causes a classifier to concentrate on the core of a cluster which contains important information.

1. Objective

In this thesis, I aim to design the combination algorithm of DBSCAN and SMOTE to strengthen a minority class distribution by over-sampling this class. This affect will guarantee the minority class detection to be satisfactory; in addition, the classifier is guided to emphasize more dense regions for the minority instances. My research shows

the improvement of the predictive performance of a classifier for both minority and majority instances in an imbalanced dataset.

2. Scope of Work

Due to the scope of my research, my framework applies a density-based concept for handling the class imbalanced problem and is operates on multiple minority class datasets with continuous attributes. In my experimental design, my framework is compared with various over-sampling techniques in the SMOTE family by evaluating accuracy, F-value (as the term of Precision and Recall), and AUC of C4.5, Ripper, and SVM available in WEKA 3.6.5 on UCI datasets.

3. Expected Outcome

After applying my framework on imbalanced UCI datasets with multiple minority classes, I expect to achieve the significant improvement of accuracy, F-value, and AUC when applying decision tree C4.5, RIPPER, and multilayer perceptron.

.

4. Research Methodology

In this thesis, the concept of my framework is to emphasize on the core information contained in the core of a cluster than the border information contained in the border of a cluster. The key idea of my approach can be acquired by defining a new data structure for the over-sampling purpose. My framework applies this data structure for generating synthetic instances into the line segments close to the core with a higher rate and the line segments far from the core at a lower rate.

My framework applies a particular graph whose shortest paths reside within the shape of a cluster. An edge between two nodes in the graph exists if and only if these

two nodes lie within a threshold distance so this graph would consist of many short edges. As an over-sampling process generates a synthetic instance and then positions it along a shortest path searched in the graph transformed from an arbitrarily shaped cluster, it will be located inside the shape of the graph because this shortest path is similar to a skeleton path, a path which is formed inside a cluster. Consequently, the density of the synthetic instances is dense nearby the core and is sparse far from the core and then a classifier is induced to emphatically learning in the important information contained around the core so the overlapping problem between a minority class and a majority class would be treated.

My framework outperforms SMOTE and Borderline-SMOTE due to the following facts. SMOTE is negatively impacted by the over-generalization problem because SMOTE blindly generalizes throughout a minority class without considering a majority class, especially in an overlapping region, which is a mix between a minority class and a majority class, so it is consequently difficult for a classifier to accurately detect an instance; however, my framework treats each region differently. Borderline-SMOTE operates only on borderline instances in the overlapping region where synthetic instances are most dense so the rate of detection of majority classes is disappointing because the classifier mis-detects instances as being positive in this context; however, my framework avoid generating positive instances around border region and will relief the negative impact around border region.

CHAPTER II

Background

In this chapter, the backgrounds of my thesis comes from my research interest, a specific problem I encounter with, suitable methods and evaluators for the class imbalance problem, experimental classifiers, and the concept of k nearest neighbours.

Data mining is one research area of Computer Science, Computer Engineering, and Information Technology, and is a process of analyzing collections of data. The objective of data mining is to discover knowledge, relations, or patterns from large databases in structures that human can understand.

Class imbalanced problem is an interesting one among classification tasks in data mining and occurs in an application when a target class (minority class) has a very small fraction compared with another class (majority class); as a result, a classifier loses its predictive performance because a huge majority class dominates a tiny minority class during classification processing.

Re-sampling techniques are categorized in the data level concept and are applied for handling the class imbalanced problem. In addition, the techniques re-balance classes in an imbalanced dataset by inserting (over-sampling) and/or deleting (under-sampling) instances into/from this dataset until the classes are approximately balanced.

Accuracy is a traditionally measure applied in a balanced dataset but inappropriate for an imbalanced dataset because this performance measure tends to count a large number of instances in a majority class as correctly classified. Fortunately, F-value and AUC are suitable for the class imbalanced problem because these performance measures concentrate on a minority class with its high priority.

In my experiment, I apply three kinds of classifiers. The first one is a statistical classifier C4.5 applied for generating a decision tree. The second one is a rule-based

classifier RIPPER applied for discovering a rule set. The last one is a neural network model MLP applied for distinguishing a set of instances which is not linearly separable.

K nearest neighbours is the concept of finding the k nearest instances which orbit around a considered instance by calculating from a distance metric such as normalized euclidean distance function which is applied in this thesis. Besides, the nearest instances are more similar to the considered instance than the other instances due to the distances between them. In this thesis, I apply k nearest neighbours to select a suitable instance to be over-sampled.

1. Data Mining

Data mining or knowledge discovery in large databases (KDD) is a challenging field of Computer Science, Computer Engineering, and Information Technology, and is a process of analyzing a collection of huge number of instances by applying various machine learning and artificial intelligence algorithms. Technically, the objective of data mining is to discover or extract knowledge, relations, or patterns from different perspectives and summarizing them into useful structures that human can understand. Data mining tool such as WEKA, RapidMiner, SASEM, or R, is one of analytical softwares for analyzing large databases, and allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. They support the main functionalities of data mining which are classification, clustering, and association analysis.

Fig 2.1 illustrates the process of data mining which starts from understanding of the application domain, the relevant prior knowledge, and the goals of the end-user. In the selection step, the process creates a target dataset by integrating a DBMS (Database Management System), or focusing on a subset of variables or data samples, on which discovery is to be performed. In the preprocessing step, the process cleans noise or outliers, collects necessary information to model, strategies for handling

missing data fields, and accounts for time sequence information and known changes. In the transformation step, the process finds useful features to represent the data depending on the goal of the task, and uses dimensionality reduction to reduce the effective number of variables under consideration or to find invariant representations for the data. In the data mining step, an analyst chooses the data mining task by deciding whether the goal of the data mining process is classification, clustering, association analysis, etc; in addition, he/she chooses the data mining algorithms by selecting methods to be used for searching for patterns in the data, deciding which models and parameters may be appropriate, and matching a particular data mining method with the overall criteria of the data mining process. In the interpretation/evaluation step, the process searches for patterns of interest in a particular representational form or a set of such representations as classification rules or trees, clustering models, association rules, and so forth. Eventually, an analyst interprets mined patterns and consolidates discovered knowledge.

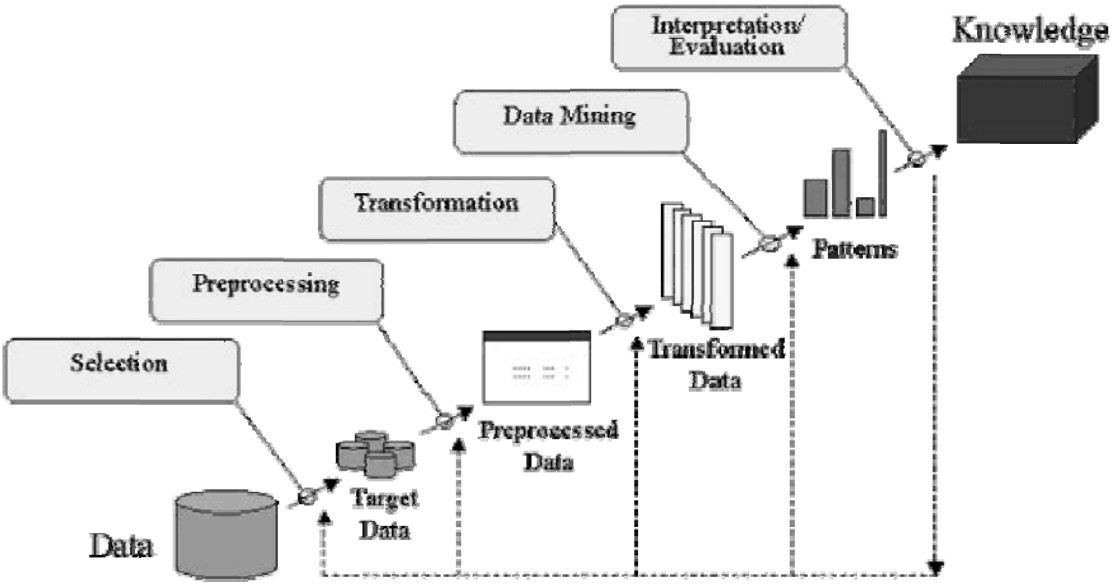

Fig. 2.1. Data mining process.

Classification, prediction, and forecasting have the similar meaning in data mining. It is a method for predicting future values by considering probability of past events. For example, I may aim to predict that a new customer will pay the bill within 60 days or take longer than 60 days. For another example, the forecast may be used to predict tomorrow weather that it will be rain, sunny, or cloudy. For the classification, I may want to diagnose whether a patient will have cancer in his/her body.

Clustering is a method for grouping data into distinct clusters that have similar characteristics so I may treat each cluster differently. For example, a health-insurance company may discover a cluster of customers who are television science-fiction fan. Consequently, the company can target this kind of customers by using television advertisements in new science-fiction episodes. The well-known and widely-used clustering algorithms are k-means and DBSCAN.

Association analysis is a method for discovering interesting relations between items in large databases. This method can be applied to market basket analysis. For example, the relation $\{Milk, Bread\} \rightarrow Butter$ means that if customers buy milk and bread together, they are likely to also buy butter. This information can be used as the basis for decisions about marketing activities such as promotional pricing or product placements.

For evaluating the performance of data mining models, separating data into training and testing sets is an important part of evaluating data mining models. Typically, when users partition a data set into a training set and testing set, a large proportion of data is used for training, and a smaller portion of the data is used for testing. Stratified sampling randomly samples the data to help ensure the proportion of class instances. By using similar data for training and testing, users can minimize the effects of data discrepancies and help classifier recognize the characteristics of the datasets. After a model has been processed by using the training set, users test the model by making predictions against the test set. Because the data in the test set already contains known

values for the class attribute, it is easy to determine whether the model's predictions are correct.

The other method evaluating the performance of a classifier is cross validation which reserves a portion of the data to test the accuracy of the model building from the set of the data. Cross validation randomly divides data into two or more subsets; training samples (used to construct the model), validated samples (some methods need these to tune the model), and test samples (evaluate performance of the model). The procedure of k-fold cross validation begins to divide the data in k parts; (k - 1) parts for building (train) and one part for predicting (test); after that the process fit the model on the training data; finally, it measures the data in the test sample. This procedure is repeated multiple times, each time dividing the data into subsets at random.

2. Class Imbalanced Problem

A dataset is considered to be imbalanced if the target class has a small number of instances compared to the other classes. A problem encountered with an imbalanced dataset is called class imbalanced problem [1], [2], [3], [4]. Many applications in these problems consider the two-class case [5], [6], [7], [8], [9], [10]. In this case, the smaller class is called the minority class of which the instance in this class is referred as positive, and the larger class is called the majority class of which the instance in this class is referred as negative.

Because the objective of the class imbalance problem is to correctly classify the (minority class) positive instances, if a dataset has more than two classes, the target class will be selected as the minority class while the remaining classes will be merged as the majority class. In multiple class datasets, there are two widely used techniques; One-Against-All (OAA) and One-Against-One (OAO). OAA treats these datasets as a binary classification problem for each distinct class so a classifier for each class is trained to predict whether the label is the class or not the class. OAO extracts all pairs of

classes and construct a binary classification between the two classes in each pair so the training set contains only elements of two classes and the other training instances are ignored during the construction.

In this thesis, I apply OAA to treat an imbalanced dataset with multiple minority classes because my framework is design to handle the two-class case. For example, a dataset in Fig 2.2 has a majority class A and two minority classes: B and C. I begin to consider B as a minority class and merge both A and C as a single majority class and then I can apply my framework to this dataset. I then repeat this framework again to complete the strategy. As a result, a classifier can generate the decision boundary among A, B, and C. OAA is suitable for my framework because it operates on a dataset as the view of a considered class and unconsidered classes.

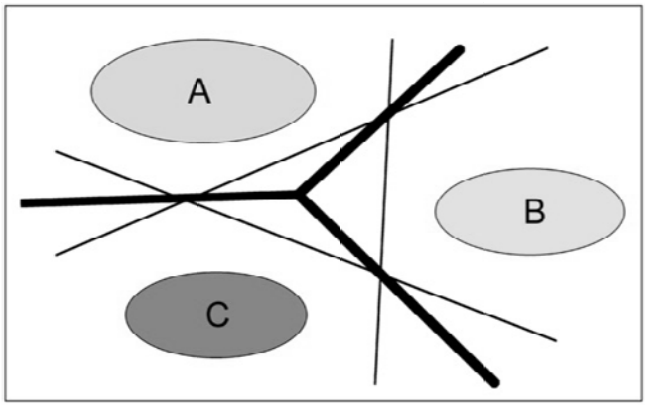

Fig. 2.2. One-Against-One.

In various real-world domains, analysts encounter many classification tasks related to class imbalanced problems, such as telecommunications risk management [11], the detection of unknown and known network intrusions [12], the detection of fraudulent telephone calls [13], in-flight helicopter gearbox fault monitoring [14], the detection of oil spills in satellite radar images [15], the identification of likely buyers of certain products in direct marketing problems [16], and the detection of

microcalcifications in mammography [17]. In these domains, a standard classifier needs to accurately predict an important and rare minority class, but the classifier seldom predicts this class due to its tiny size.

The example of an application which encounters the class imbalance problem is network intrusion detection. Intrusion occurs when hackers or virus attack the machine in a computer network. Number of intrusions on the network is typically a very small fraction of the total network traffic and is more important than typical usages. If an administrator can built a classifier which efficiently detects the intrusion before it compromises the network, the network will be more secured.

There was a study [18] which evaluated AUC of classifiers training after these preprocessing techniques, over-sampling, under-sampling, and data cleaning, by applying decision trees C4.5. The experiments showed that the over-sampling techniques were better than the under-sampling techniques. Moreover, the combinations of over-sampling and data cleaning provided satisfactory results when a minority class was small.

In this thesis, the density-based minority over-sampling framework for handling the class imbalanced problem is proposed. My framework relies on the density concept of arbitrarily shaped clusters of positive instances. In addition, my framework considers the density of each region in an imbalanced dataset and then generates more synthetic instances in the dense regions rather than the sparse regions because the information contained in the dense regions is more important than that in sparse regions.

3. Re-sampling Technique

One strategy for handling class imbalanced problems is a re-sampling technique [3], [19]. It is a preprocessing technique that adjusts the distribution of classes in an imbalanced dataset until all classes are nearly balanced before feeding this modified dataset into a classification algorithm.

There are two types of re-sampling techniques: over-sampling techniques and under-sampling techniques. The former inserts positive instances into a minority class, while the latter removes negative instances from a majority class. Both techniques change the distribution of a dataset until its classes are approximately equally represented. However, the over-sampling technique may encounter the over-fitting problem [20] if this technique creates smaller and more specific decision regions by duplicating instances. In contrast, the under-sampling technique may diminish some important information in a dataset-especially in its core.

Fig 2.3 illustrates an original dataset (left side) and a modified dataset (right side) in the two dimensional space after applying both over-sampling and under-sampling techniques. In this figure, a symbol + represents a positive instance in a minority class and a symbol - represents a negative instance in a majority class. Over-sampling duplicates or synthesizes positive instances into a minority class and under-sampling cleans some negative instances from a majority class; as a result, a minority class is better learned by classifiers.

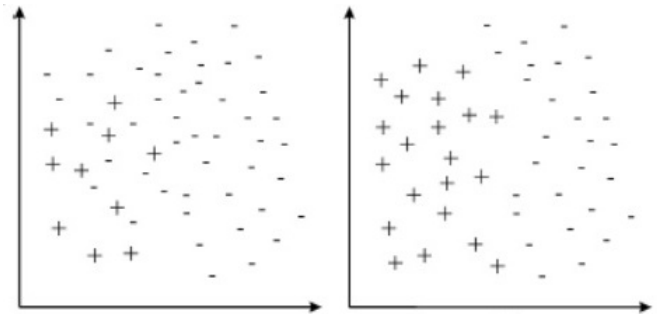

Fig. 2.3. Over-sampling and Under-sampling.

SHRINK [10] is a system which searches for the best positive region. The region is an overlapping region, mixing of positive instances and negative instances, which have a maximum ratio of the positive instances to the negative instances. The system insists that the overlapping region be classified as positive, whether positive instances

prevail in the region or not. However, the system fail to learn on disjunctive concepts [4] because the system was designed specifically for overlapping classes so there is no benefits if the classes do not overlap.

One-Sided Selection [19] considers only numeric attributes. the heuristic technique under-samples a majority class by eliminating the negative instances, which can easily be detected using the concept of Tomek Links [18], from noise regions and borderline regions; however, the experiment reveals that the performance of the induced classifier is largely unaffected by the choice of removed negative instances. In addition, negative instances can roughly be divided into four regions: noise, borderline, safe, and redundant. The noise region overlaps the decision regions of a minority class. The borderline region is the boundary between positive and negative regions and is unreliable due to the fact that even a small amount of noise instances can send the borderline instances to the wrong side of decision surface. The safe region is kept for future classification tasks. The redundant instances do not harm correct classifications but increase classification costs.

Fig 2.4 illustrates an original dataset (left side) and a modified dataset (right side) after applying One-Sided Selection which detects Tomek Links to remove both noise and borderline instances in a majority class; as a result, a minority class is more dominant to be recognized by classifiers. Note that Tomek Links connect between positive and negative instances which are nearest neighbours to each other.

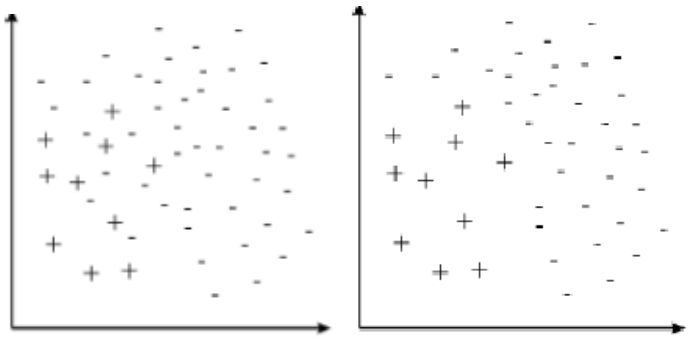

Fig 2.4. One-Sided Selection.

Another strategy is to apply cost-sensitive learning, which operates on an imbalanced dataset by assigning distinct costs to correctly classified instances or classification errors [21], [22], [23]. Other techniques deal with this situation differently, such as internal bias discrimination, boosting based algorithm [24], [25], and clustering based classification [2].

4. Performance Measure

The performance of a classifier is customarily evaluated by a confusion matrix, as shown in Table 2.1. The rows of the table are the actual class label of an instance, and the columns are the class labels predicted by a classifier. Typically, the class label of an instance in a minority class is set as positive and that of a majority class is set as negative. TP, True Positive, is the number of positive instances correctly classified. FN, False Negative, is the number of positive instances incorrectly classified. FP, False Positive, is the number of negative instances incorrectly classified. TN, True Negative, is the number of negative instances correctly classified. From Table 2.1, the six performance measures of classification [9], accuracy, Precision, Recall, F-value, TP rate, and FP rate, are defined by formulae (2.1) through (2.6).

Table 2.1. A confusion matrix for the two-class imbalance problem.

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

$$\text{Accuracy} = (TP + TN) / (TP + FN + FP + TN) \quad (2.1)$$

$$\text{Recall} = TP / (TP + FN) \quad (2.2)$$

$$\text{Precision} = TP / (TP + FP) \quad (2.3)$$

$$\text{F-value} = ((1 + \beta)^2 \cdot \text{Recall} \cdot \text{Precision}) / (\beta^2 \cdot \text{Recall} + \text{Precision}) \quad (2.4)$$

$$\text{TP Rate} = \text{Sensitivity} = TP / (TP + FN) \quad (2.5)$$

$$\text{FP Rate} = 1 - \text{Specificity} = FP / (TN + FP) \quad (2.6)$$

In the class imbalance problem, accuracy is an inappropriate measure due to the tiny misclassification error on a minority class. In the domain studied by Lewis and Catlett [26], their dataset had only 0.2% positive instances and nearly 100% negative instances. A trivial classifier can reach an accuracy of 99.8% by predicting every instance as a negative instance and ignoring the existence of the positive instances. However, the objective of the problem is to aim for high prediction performance on a minority class.

Considering the definition of accuracy, if most positive instances are misclassified and most negative instances are correctly classified by a classifier, accuracy will be still high because the large number of these negative instances can influence the whole classification result on accuracy. On the other hand, Precision and Recall are effective measures for the problem because they can evaluate the classification rates by concentrating on a minority class.

F-value [27] integrates recall and precision. The F-value is large when both recall and precision are large. The parameter β, corresponding to the relative importance of precision and recall, is usually set to 1, meaning that Precision is as important as Recall.

ROC [28], the Receiver Operating Characteristic, is a standard technique for summarizing the prediction performance of a classifier over the range of trade-offs between TP rate and FP rate. The ROC curve is a graph in two-dimensional space in which the x-axis represents FP rate and the y-axis represents TP rate. One ROC curve

would be considered to dominate other ROC curves if the one is always above and to the left of the others.

AUC [28], Area Under ROC, can also be applied to evaluate the performance of a classifier. The AUC of a classifier is equivalent to the probability that this classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. If one AUC is the largest, its ROC would dominate other ROC curves on some regions.

Fig 2.5 illustrates ROC curves of two strep rules: VA and NE. In this graph, the predictive performance of VA is better than that of NE. Note that at the ideal point, all positive instances are correctly classified and no negative instances are misclassified as positive instances.

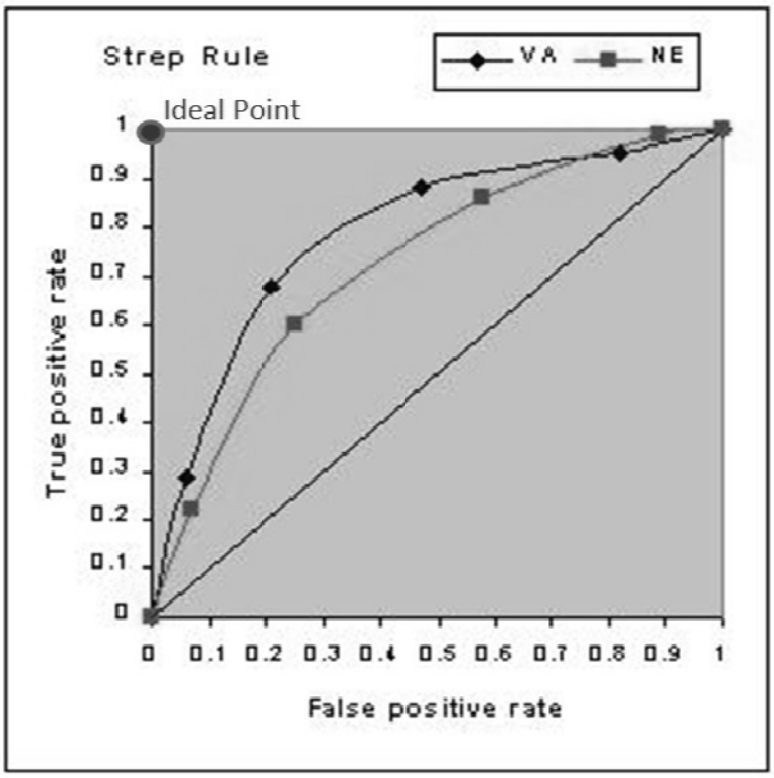

Fig 2.5. ROC curve.

5. Experimental Classifier

C4.5 [29] is an algorithm used to generate a decision tree developed by Ross Quinlan and is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification, and for this reason, it is often referred to as a statistical classifier.

Fig 2.6 illustrates the example of a decision tree involving the decision to play or not play based on climate conditions. In this case, outlook is in the position of the root node. The degrees of the node are attribute values. In this example, the child nodes are tests of humidity and windy, leading to the leaf nodes which are the actual classifications. This example also includes the corresponding data, also referred to as instances. In my example, there are 9 *"play"* days and 5 *"no play"* days.

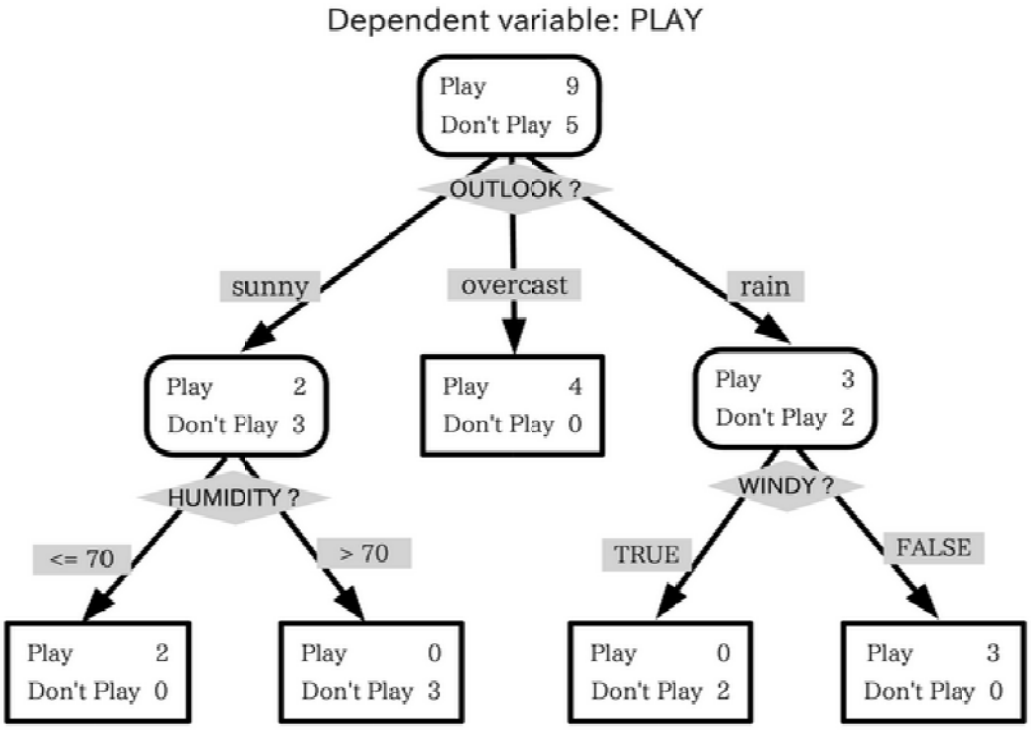

Fig 2.6. Decision tree.

RIPPER [30] is a rule-based classifier is based on if and then conditions that called the rule set. The rules have to have two properties. The first one is that the rules should be mutually exclusive. The second one is that the rules should be exhaustive. The rule based classifier is very similar to the tree based classifier and it is very easy to convert a tree to a rule based classifier. Actually in order to guarantee the above two properties it is recommended to construct a tree and then convert it to a set of rules. However, rule base classifier has advantage over decision tree that its rules can be simplified.

A rule r covers an instance x if the attribute of the instance satisfy the condition of the rule. For example, a rule r and instances x_1, x_2, and x_3 are given as follows.

$$r\text{: } (Age < 35) \wedge (Status = Married) \rightarrow Cheat = No$$

$$x_1\text{: } (Age = 29, Status = Married, Refund = No)$$

$$x_2\text{: } (Age = 28, Status = Single, Refund = Yes)$$

$$x_3\text{: } (Age = 38, Status = Divorced, Refund = No)$$

To consider all of them, only x_1 is covered by the rule r. Note that more than one rule may cover the same instance.

MLP [27] (multilayer perceptron) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate output. MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron or processing element with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network. MLP is a modification of the standard linear perceptron, which can distinguish data that is not linearly separable.

Fig 2.7 illustrates MLP. In brief, an input vector is placed on the input nodes and is propagated to the output layer via the weight connections and the hidden-layer. This is done for each vector in the training set (one iteration). Each node in the hidden and

output layers transforms the sum of its inputs via an activation function, normally known as a sigmoid function.

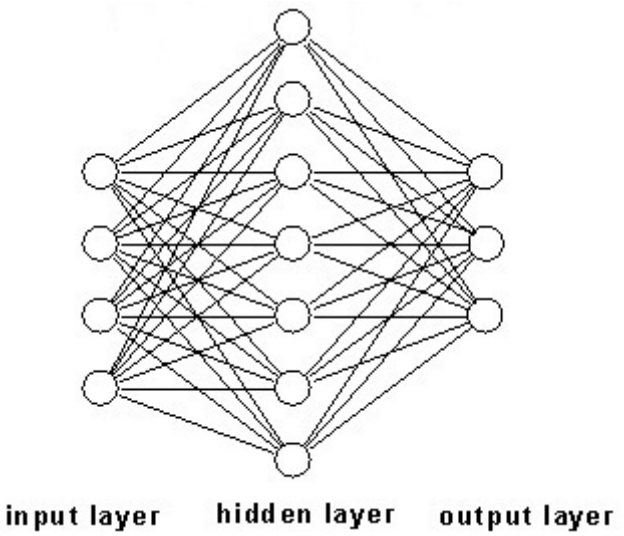

Fig 2.7. A simplified MLP network architecture.

6. K Nearest Neighbours

To demonstrate the k nearest neighbors (KNN) analysis, consider the task of classifying a new object (query point) among a number of known examples. This is shown in Fig 2.8, which depicts the examples (instances) with the plus and minus signs and the query point with a circle. The task is to estimate (classify) the outcome of the query point based on a selected number of its nearest neighbors. In other words, I want to know whether the query point can be classified as a plus or a minus sign. To proceed, consider the outcome of KNN based on 1 nearest neighbor. It is clear that in this case KNN will predict the outcome of the query point with a plus (since the closest point carries a plus sign). Now I increase the number of nearest neighbors to 2, i.e., 2 nearest neighbors. This time KNN will not be able to classify the outcome of the query point since the second closest point is a minus, and so both the plus and the minus signs achieve the same score (i.e., win the same number of votes). For the next step, I

increase the number of nearest neighbors to 5 (5 nearest neighbors). This will define a nearest neighbor region, which is indicated by the circle shown in the figure. Since there are 2 and 3 plus and minus signs, respectively, in this circle KNN will assign a minus sign to the outcome of the query point.

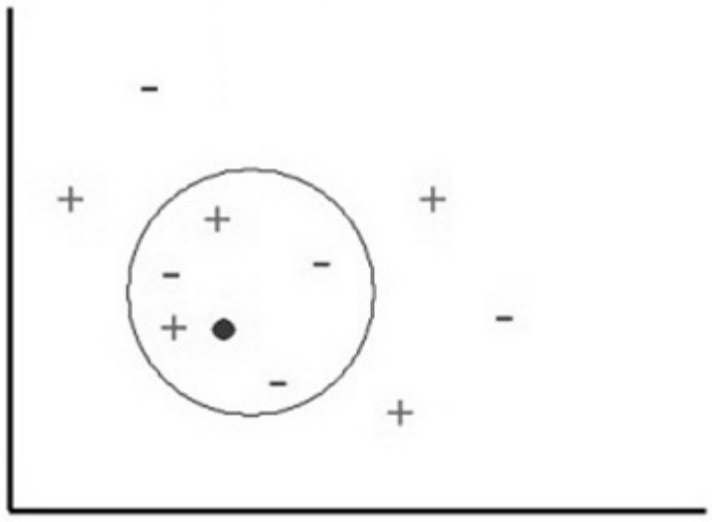

Fig 2.8. K nearest neighbors.

I apply this concept for determining the k nearest neighbours of an instance and then use them in the over-sampling step for generating synthetic instances because the considered instance and its selected k nearest neighbours are considered similar. In this thesis, I fix k as 5 as the default value.

CHAPTER III

Related Work

In this chapter, my thesis relates to two researches; a widely-used re-sampling technique, SMOTE (Synthetic Minority Over-sampling TEchnique) [8] and a well-known clustering algorithm, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [31] because I aim to combine them to achieve a new efficient over-sampling framework.

SMOTE is the state-of-the-art over-sampling technique which generates a synthetic instance along the line segment between each instance and its selected nearest neighbor from a minority class. Unfortunately, SMOTE blindly generalizes the regions of a minority class without considering a majority class; thus, SMOTE encounters the overlapping problem.

DBSCAN is a density-based clustering algorithm which is applied to discover clusters of arbitrary shapes and also to detect noises. In the algorithm, for each instance in a cluster, the neighbourhood of this instance within the radius Eps has to contain at least $MinPts$ instances. To approximate the values of Eps and $MinPts$, a sorted k-dist graph is considered for this purpose.

1. SMOTE

SMOTE is the state-of-the-art over-sampling technique which generates synthetic instances by operating in the feature space rather than the data space. These synthetic instances are generated along the line segments joining each instance to its k nearest neighbours. A dataset can have continuous attributes or nominal attributes. The authors chose the euclidean distance metric for continuous attributes.

By the SMOTE algorithm, for each instance, compute the vector difference between the feature vector of the instance and one from its k nearest neighbours and

then multiply this difference by a random number between 0 and 1. After that, add this difference to the feature vector of an original feature vector, thus generating the new synthetic instance. This process is illustrated in Fig 3.1 in which a white point and a gray point represent a positive instance and a synthetic instance, respectively; in addition, a centre point is a considered positive instance and satellite points orbited around it are its k nearest neighbours

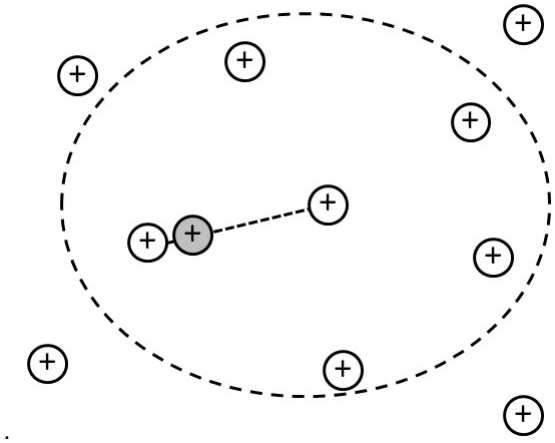

Fig 3.1: SMOTE over-sampling.

The synthetic instances cause a classifier to create larger and less specific decision regions rather than smaller and more specific regions. More general regions are learned for positive instances rather than those positive instances being subsumed by negative instances around them. The effect is that a classifier generalizes better.

However, SMOTE encounters the overgeneralization problem because this technique blindly generalizes the region of a minority class without considering a majority class-especially in overlapping regions. If a dataset has a highly skewed class distribution, the synthetic instances in overlapping regions will be sparse with respect to the instances in a majority class. Thus, the majority and minority classes would be blended.

2. DBSCAN

DBSCAN is the density-based clustering algorithm which is applied to discover clusters of arbitrary shapes based on the metric distance and number of neighbor points. In addition, DBSCAN can be used as a noise detection algorithm.

The algorithm requires two parameters; the distance Eps and the threshold $MinPts$. The key idea of DBSCAN is that for each instance in a cluster, its neighbourhood within the radius Eps must include at least the number of instances $MinPts$. Their definitions are repeated for the purpose of comparison.

Definition 1: (Eps-neighbourhood)
Let D be a dataset. The Eps-neighbourhood of an instance p, denoted by $N_{Eps}(p)$, is defined by $N_{Eps}(p) = \{q \in D \mid \text{dist}(p, q) \leq Eps\}$.

Definition 2: (Directly density-reachable)
An instance p is directly density-reachable from an instance q wrt. Eps and $MinPts$ if

1) $p \in N_{Eps}(q)$ and
2) $|N_{Eps}(q)| \geq MinPts$ (Core instance condition).

By Definition 1, the Eps-neighbourhood of an instance p is the set of all instances whose distances measured from p do not exceed the threshold Eps. In addition, the metric function dist(p, q) returns the distance between the instances p and q. Any type of distance can be applied to this function. In this thesis document, I illustrate all figures in the two-dimensional space with normalized euclidean distance function.

In a cluster, there are two types of instances; core instances and border instances. The first type relies on the core instance condition and is located in the center

of the cluster. The second type does not rely on this core instance condition and is located around the border of the cluster. Fig. 3.2 (a) illustrates where a core instance and a border instance are located in the two dimensional space. In this figure, each point represents an instance and each circle represents the region of the radius Eps from a center point. If there are points located in this space, they will be in the Eps-neighbourhood of the center point. This example uses a $MinPts$ value of 5; thus, instance p is a border instance while instance q is a core instance.

By Definition 2, if an instance p is a member of the Eps-neighbourhood of a core instance q, p will be directly density-reachable from q. This relation is not symmetric when one is a core instance and the other is a border instance. Fig. 3.2 (b) illustrates the asymmetric case in which an arrow represents the direction of directly density-reachable relation; a head point is directly density-reachable from a tail point. In this figure, the instance p is directly density-reachable from the instance q, but q is not directly density-reachable from p.

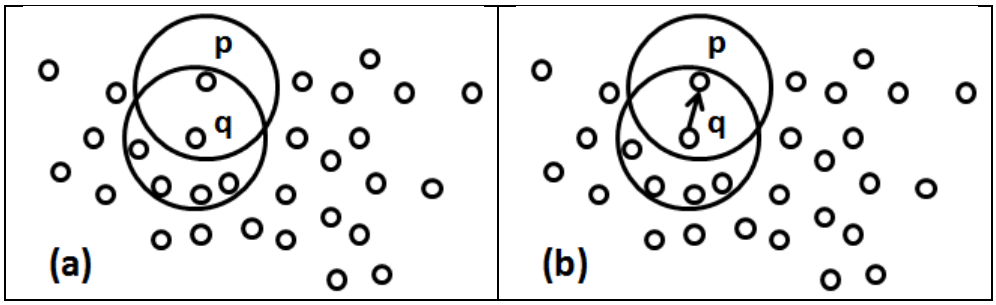

Fig. 3.2. (a) Core instance and border instance (b) Directly density-reachable.

Definition 3: (Density-reachable)

An instance p is density-reachable from an instance q wrt. Eps and $MinPts$ if there is a chain of instances $p_1, \ldots, p_n, p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

Definition 4: (Density-connected)

An instance p is density-connected to an instance q wrt. Eps and $MinPts$ if there is an instance r such that both p and q are density-reachable from r wrt. Eps and $MinPts$.

By Definition 3, density-reachable is the transitive extension of directly density-reachable. In addition, density-reachable is transitive but not symmetric because if there exists at least one border instance in a pair, this border instance cannot hold the core instance condition and thus the partner cannot be density-reachable from the border instance. Fig. 3.3 (a) illustrates the asymmetric case. In this figure, the instance p is density-reachable from the instance q, but q is not density-reachable from p.

By Definition 4, if there is a core instance from which two instances in a cluster are density-reachable, these instances will be density-connected to each other. This relation is not only symmetric but also reflexive. Only a pair of core instances can hold the reflexive property. Fig. 3.3 (b) illustrates the symmetric case. In this figure, the instances p and q are density-connected to each other by an instance r.

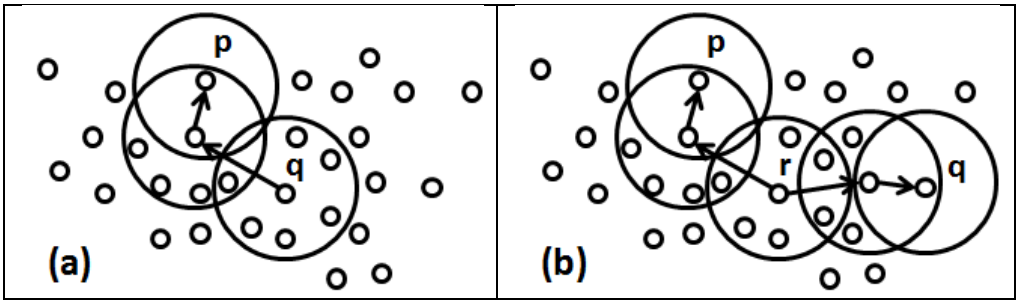

Fig. 3.3. (a) Density-reachable (b) Density-connected.

Definition 5: (Cluster)

A cluster C wrt. Eps and $MinPts$ is a non-empty subset of a dataset D satisfying the following conditions:

1) $\forall$ p, q: if $p \in C$ and q is density-reachable from p wrt. Eps and $MinPts$, then $q \in C$ (Maximality)

2) $\forall\, p, q \in C$: p is density-connected to q wrt. Eps and $MinPts$ (Connectivity)

Definition 6: (Noise)

Let $C_1, \ldots, C_k$ be the k clusters of a dataset D wrt. Eps_i and $MinPts_i$ where $i \in \{1, \ldots, k\}$. $noise$ is the set of instances not belonging to any clusters C_i and is defined as $noise = \{p \in D \mid \forall i: p \notin C_i\}$.

By Definition 5 and Definition 6, a density-based cluster is defined as a set of instances that satisfy the density-connected relation and are maximal with respect to density-reachable. A cluster includes not only core instances but also border instances. *Noise* is a set of instances that are not located in any clusters. Note that some border instances are treated as part of a cluster, while the remainder is considered to be noise. A core instance cannot be noise because a cluster must include at least one core instance to satisfy the maximality and connectivity conditions restricted by Definition 5. Moreover, only one core instance and its Eps-neighbourhood can be treated as the thinnest cluster.

DBSCAN can not only discover arbitrarily shaped clusters but also detect noise instances in a dataset. This algorithm constructs a cluster by determining a core instance as a root, then retrieving all instances that are density-reachable from this root to acquire a cluster included in this root. The time complexity of the algorithm DBSCAN was analyzed as a logarithmic function $O(n\lg n)$ [31], where n is the number of instances in a dataset.

The DBSCAN algorithm operates in the following manner. For each instance in a dataset, determine whether this instance satisfies the core instance condition. If an instance does, it will be a core instance p. After that, construct a cluster C by including all instances in the Eps-neighbourhood of p. For each instance in C that has not yet been processed, seek a core instance q and then merge C with the instances in the

Eps-neighbourhood of q that are not already located in C. Recursively, check their Eps-neighbourhood in the next step. This loop executes iteratively until no new instance can be included in the current cluster C.

However, the appropriate values of Eps and $MinPts$ are difficult to decide. Fortunately, the authors provide a heuristic to determine these parameters from the given k**,** the number of nearest neighbours. Let *d* be the distance between an instance p and its furthest kth nearest neighbour. If Eps is assigned as d**,** the number of instances in the Eps-neighbourhood of p will be exactly (k + 1). In cases where the distances between several instances and p are the same, the Eps-neighbourhood of p would include more than (k + 1) instances.

The function k-dist requires an instance p and returns the distance d. The sorted k-dist graph illustrated in Fig. 3.4 plots all distances d in descending order. In this figure, the *x*-axis represents each instance p in a dataset and, the *y*-axis represents the distance d assigned by k-dist(p). This graph can guide the values of Eps and $MinPts$ of the thinnest cluster. If an instance q is selected as the threshold instance, then setting Eps and $MinPts$ to k-dist(q) and k, respectively, any instances p for which k-dist(p) does not exceed k-dist(q) will be core instances and the remainder will be border instances.

In the sorted k-dist graph, all instances that lie to the right of the threshold instance are core instances, whereas all instances that lie to the left of the threshold instance are border instances. Note that the threshold instance is considered to be a core instance. In addition, the threshold instance should be the first instance in the first valley of the sorted k-dist graph. It is inconvenient to design an algorithm that can automatically detect this threshold instance; however, an analyst can conveniently determine this threshold instance by visualizing and considering this sorted k-dist graph.

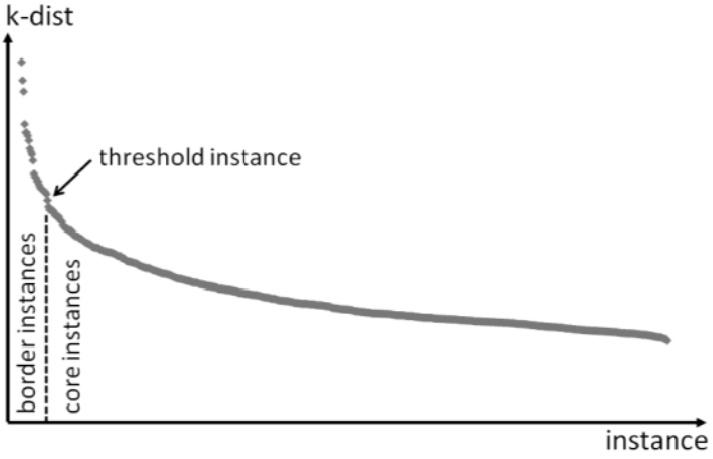

Fig. 3.4. A sorted k-dist graph.

CHAPTER IV

Problem Methodology

In this chapter, I describe my three approaches to handle the class imbalanced problems, two over-sampling and one under-sampling techniques, as follows.

Safe-Level-SMOTE [7] is an improvement of SMOTE which carefully over-samples a minority class by generating synthetic instances along the same line segment of SMOTE with different weight degree called safe level computed by counting the minority class nearest neighbours. By synthesizing instances more around larger safe level instances, these synthetic instances are located closer to minority instances than majority instances.

MUTE (Majority Under-sampling Technique) [6] is an under-sampling technique which gets rid of noise majority instances which overlap with minority instances. The removal majority instances are considered based on their safe levels relying on the Safe-Level-SMOTE concept. MUTE not only reduces the classifier construction time because of a downsizing dataset but also improves the prediction rate on a minority class.

DBSMOTE (Density-Based Minority Over-sampling Technique) [9] relies on a density-based notion of clusters and is designed to over-sample an arbitrarily shaped cluster discovered by DBSCAN. DBSMOTE generates synthetic instances along a shortest path from each positive instance to a pseudo-centroid of a minority-class cluster. Consequently, these synthetic instances are dense near this centroid and are sparse far from this centroid.

1. Safe-Level-SMOTE

SMOTE is an original work of the over-sampling techniques for handling the class imbalanced problem by generating synthetic instances. Unfortunately, SMOTE might

generates synthetic instances to be crashed against the opposite-class instances especially in the border of a minority class; thus, SMOTE encounters the overlapping problem.

Basing on SMOTE, Safe-Level-SMOTE carefully over-samples by considering safe positions to generate synthetic instances. In addition, Safe-Level-SMOTE assigns each positive instance its safe level before generating synthetic instances along the line between the positive instance and their selected positive nearest neighbours. Each synthetic instance is positioned closer to the largest safe level so all synthetic instances are generated only in safe regions. This is the advantage of my technique because it can prevent the case of over-sampling in unwanted locations such as noise and overlapping regions.

The safe level (sl) is defined as formula (4.1). To interpret this formula, if the safe level of an instance is close to 0, the instance is nearly noise. But, if it is close to k, the instance is considered safe for over-sampling.

The safe level ratio (sl_ratio) is defined as formula (4.2). The safe level ratio is used for selecting the safe positions in an over-sampling line segment of SMOTE to carefully generate synthetic instances of a minority class.

safe level = the number of positive instances among its k nearest neighbours (4.1)

safe level ratio = sl of a positive instance / sl of a nearest neighbour (4.2)

The Safe-Level-SMOTE algorithm is showed in Fig. 4.1. All variables in this algorithm are described as follows. p is an instance in the set of all original positive instances D. n is a selected nearest neighbours of p. s included in the set of all synthetic positive instances D' is a synthetic instance. sl_p and sl_n are safe level of p and safe level of n respectively. sl_ratio is safe level ratio. $numattrs$ is the number of attributes. dif is

the difference between the values of n and p at the same attribute id. gap is a random fraction of dif. $p[i]$, $n[i]$, and $s[i]$ are the numeric values of the instances at i^{th} attribute. In the algorithm, D and D' are set of positive instances. p, n, and s are vectors. sl_p, sl_n, sl_ratio, $numattrs$, dif, gap, $p[i]$, $n[i]$, and $s[i]$ are scalars.

The algorithm begins to computer the k nearest neighbours of p. After that, the algorithm assigns the safe level to p and the safe level to n and then calculates the safe level ratio of the pair of p and n. There are five cases corresponding to the value of safe level ratio illustrated in Fig. 4.2. In this figure, the meanings of all variables are described below.

The first case illustrated in Fig. 4.2 (a). The safe level ratio is equal to ∞ and the safe level of p is equal to 0. It means that both p and n are noise instances because all k nearest neighbours of p and n are opposite-class instances. If this case occurs, synthetic instance will not be generated because the algorithm does not want to emphasize the important of noise regions.

The second case illustrated in Fig. 4.2 (b). The safe level ratio is equal to ∞ and the safe level of p is not equal to 0. It means that n is noise because all k nearest neighbours of n are negative instances. If this case occurs, a synthetic instance will be generated far from noise instance n by duplicating p because the algorithm want to avoid the noise instance n.

The third case illustrated in Fig. 4.2 (c). The safe level ratio is equal to 1. It means that the safe level of p and n are the same. If this case occurs, a synthetic instance will be generated along the line between p and n because p is as safe as n. Besides, this case is SMOTE.

The fourth case illustrated in Fig. 4.2 (d). The safe level ratio is greater than 1. It means that the safe level of p is greater than that of n. If this case occurs, a synthetic instance is positioned closer to p rather than n because p is safer than n. The synthetic

instance will be generated in the range [0, 1/sl_ratio] illustrated by the dark line in the figure.

The fifth case illustrated in Fig. 4.2 (e). The safe level ratio is less than 1. It means that the safe level of p is less than that of n. If this case occurs, a synthetic instance is positioned closer to n rather than p because n is safer than p**.** The synthetic instance will be generated in the range [1 - sl_ratio, 1] illustrated by the dark line in the figure.

In the algorithm, when each iteration of for loop in line 2 finishes, if the first case does not occurs, s will be generated along the specific-ranged line segment between p and n, and then s will be added to D'. After the algorithm terminates, it returns a set of all synthetic instances D'. The algorithm generates $|D|$ - t synthetic instances where $|D|$ is the number of all positive instances in D, and t is the number of iterations that satisfy the first case.

Safe-Level-SMOTE carefully over-samples a minority class in an imbalanced dataset. Each synthetic instance is generated in safe position by considering the safe level ratio of the pair of instances. The synthetic instances generated in safe positions can improve prediction performance of classifiers on the minority class.

Input: a set of all original positive instances D

Output: a set of all synthetic positive instances D'

1. $D' = \varnothing$
2. for each positive instance p in D {
3. randomly select one from the k nearest neighbours of p, call it n
4. compute sl_p and sl_n
5. if ($sl_n \neq 0$)
6. $sl\_ratio = sl_p / sl_n$
7. else
8. $sl\_ratio = \infty$
9. if ($sl\_ratio = \infty \wedge sl_p = 0$) ; the 1st case
10. skip
11. else
12. for ($atti$ = 1 to $numattrs$) {
13. if ($sl\_ratio = \infty \wedge sl_p \neq 0$) ; the 2nd case
14. $gap = 0$
15. else if ($sl\_ratio = 1$) ; the 3rd case
16. random a number between 0 and 1, call it gap
17. else if ($sl\_ratio > 1$) ; the 4th case
18. random a number between 0 and 1/sl_ratio, call it gap
19. else if ($sl\_ratio < 1$) ; the 5th case
20. random a number between 1 - sl_ratio and 1, call it gap
21. $dif = n[atti] - p[atti]$
22. $s[atti] = p[atti] + gap \cdot dif$
23. }
24. $D' = D' \cup \{s\}$
25. }
26. return D'

Fig. 4.1. Safe-Level-SMOTE algorithm.

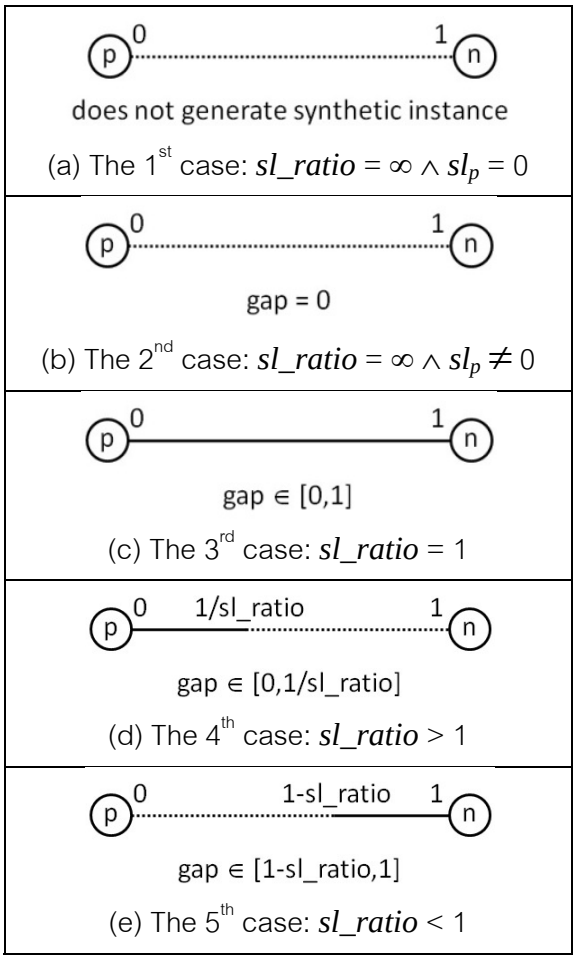

Fig. 4.2. The five cases corresponding to the safe level ratio.

2. MUTE

My proposed strategy, MUTE was inspired by Safe-Level-SMOTE, which defines safe levels on only minority (positive) instances before processing an over-sampling routine. In contrast to Safe-Level-SMOTE, MUTE applies safe levels on majority (negative) instances for the purpose of under-sampling.

A safe level of a majority instance, calculated by (4.1), is computed by the number of minority instances among k nearest neighbours. A majority instance is located in a safe region if a safe level is equal to 0. On the other hand, if a safe level is equal to k, a majority instance is considered to be noise.

The MUTE algorithm, shown in Fig. 4.3, is to wipe out the Majority Noise Set, defined by Definition 7, from a dataset. In the algorithm, τ is a minimum number of minority nearest neighbours of a majority instance, which permits MUTE to remove this instance, thus τ is set as k because I aim to delete only noise majority instances in a dataset.

Input: an original dataset D, a threshold τ

Output: an under-sampling dataset D - N

1. $N = \varnothing$
2. for each majority instance n in D
3. if $sl_n \geq \tau$
4. $N = N \cup \{n\}$
5. return D - N

Fig. 4.3 MUTE algorithm.

Definition 7: Let D be a dataset, and sl_n be a safe level of a majority instance n. The Majority Noise Set is defined by $N = \{n \in D \mid sl_n = k\}$.

In the class imbalanced problem, noise majority instances are frequently screwed into the crowd of minority instances as shown in Fig. 4.4, in which symbols + and – represent minority and majority instances respectively. MUTE cleans this crowd to be more unmixed with majority instances in an overlapping region. After cleaning this region, a classifier better generates a decision boundary because majority and minority instances are not extremely blended. In addition, MUTE does not diminish important information in a dataset because each delible majority instance is orbited by all nearest neighbors from a minority class. By-product of MUTE is the speedup for a classifier construction.

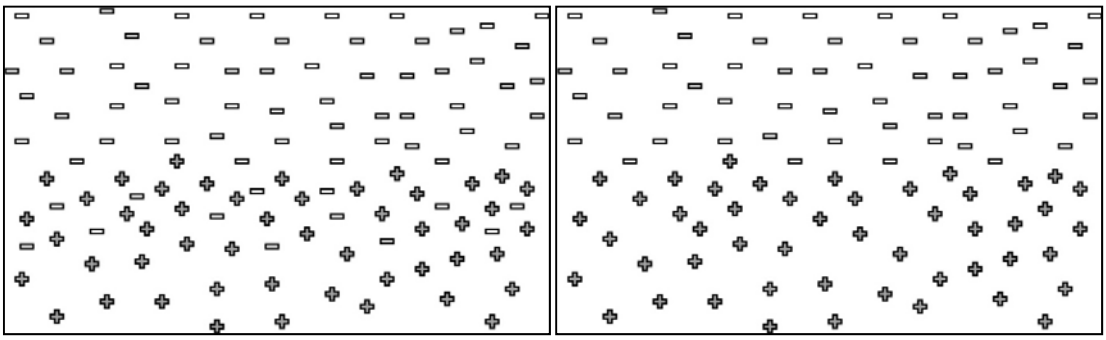

Fig. 4.4. An overlapping region before (left) and after (right) under-sampling.

3. DBSMOTE

In the class imbalanced dataset, there are two significant questions to be answered: Which minority class region should be emphasized by an over-sampling technique? and How should synthetic instances generated by the SMOTE family be distributed?

For the first question, typically, a dataset is divided into three regions; noise, safe, and overlapping. A noise region is located outside a cluster. A classifier often detects noise instances as negative because negative instances encompass these noise instances, and thus an over-sampling technique should not operate in this region. A safe region is located inside a cluster. A classifier easily recognizes this region because it has sufficient numbers of instances. However, for the class imbalance problem, a safe region of a minority class does not contain enough instances, and thus a classifier often misclassifies this region. An overlapping region is located around a cluster border, which contains a blend of positive and negative instances. This region is detected with great difficultly because a classifier cannot efficiently distinguish between the two classes, and thus over-sampling in this region might be harmful. To summarize, for the class imbalance problem, an efficient over-sampling technique should concentrate on a safe region and treat with caution any overlapping regions.

For the second question, SMOTE operates throughout a dataset, and thus synthetic instances are spread throughout every region. Borderline-SMOTE operates only

on a dataset border, and thus synthetic instances are dense in an overlapping region. Safe-Level-SMOTE operates throughout a dataset and positions synthetic instances in an overlapping region close to a safe region. Accordingly, these instances are sparse in an overlapping region and are not dense in a safe region. In this thesis, I design a new re-sampling technique which produces more synthetic instances around a dataset core than a dataset border and does not operate within a noise region. Thus these instances are dense in a safe region and are sparse in an overlapping region.

The efficient over-sampling technique called DBSMOTE is proposed for handling class imbalanced problems. This technique combines the density-based clustering algorithm DBSCAN and the over-sampling technique SMOTE.

Naturally, the distribution of any class in a real-world balanced dataset is compact near the core of a cluster and is sparse near the edge of the cluster. Consequently, the predicted result for an unidentified instance depends on its location. A classifier would predict the instance to be the same class as the core if the instance was close enough to the core.

The development of DBSMOTE was inspired by the over-sampling technique Borderline-SMOTE [9], which concentrates on over-sampling only the borderline regions of a cluster. Normally, these regions contain a high overlap between positive instances and negative instances. After over-sampling, these regions are dense in the positive instances, and thus a trivial classifier would predict more instances as positive instances whether they are positive instances or not. Accordingly, this effect improves a classifier's detection rate on a minority class but decreases it on a majority class. However, an efficient over-sampling technique should induce a classifier to accurately predict a minority class while not sacrificing the prediction rate on a majority class.

The approach of DBSMOTE is opposite to Borderline-SMOTE, which not over-samples around the center of a cluster, but DBSMOTE concentrates on this center. The concept of DBSMOTE is to emphasize the core information rather than the border information. This goal can be accomplished by over-sampling different regions with

different rates. The regions near the centroid of a cluster are over-sampled at a higher rate than the regions far from the centroid because the center of a cluster is more important than the edge of a cluster. Thus synthetic instances are dense near the centroid and sparse far from the centroid. Normally, the centroid of a cluster is located in the center of the cluster and is computed by the average of all instances.

In this thesis, I designed a new data structure called a directly density-reachable graph, which is defined in Definition 8. The graph is transformed by a cluster of instances and is an underlying weighted directed graph [32]. In the graph, each node represents an instance in a cluster while each edge represents a directly density-reachable relation. To consider a pair of connected nodes, if at least one is directly density-reachable from the other, the edge between this pair will be created. In the weight function, each weight is computed based on the distance in the pair. Note that R is a set of real numbers.

Definition 8: (Directly density-reachable graph)
A directly density-reachable graph of a cluster C discovered by DBSCAN, denoted by $G(C) = (V, E)$, where V is a set of nodes represented as instances in C and E is a set of edges. E is defined as $E = \{(v_1, v_2) \in V \times V \mid$ an instance v_1 is directly density-reachable from an instance v_2 wrt. Eps and $MinPts$ or vice versa$\}$. Let w: $E \rightarrow R$ be a weight function where $w(v_1, v_2)$ is equal to the distance between a pair of connected nodes v_1, $v_2 \in V$.

A directly density-reachable graph is illustrated in Fig. 4.5. All black points are core instances and the others are border instances. At least one from a pair of connected nodes is guaranteed to be a core instance, but no link connects a pair of border instances because they cannot hold the directly density-reachable relation in Definition 2.

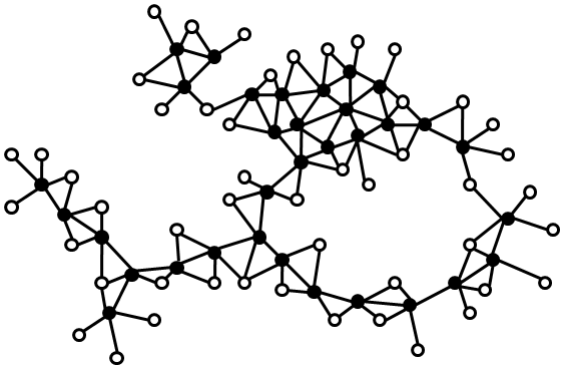

Fig. 4.5. A directly density-reachable graph.

A directly density-reachable graph was designed based on the requirements of my research, which needs a particular graph whose shortest paths reside within the shape of a cluster. In the graph, each edge exists if and only if the connected nodes in a pair are close enough to each other, and thus this graph is composed of many low-weight edges. The edges of a complete graph, whose edges link in every pair of nodes whether the distances in the pairs are high weight or low weight, do not fit the shape of a cluster; however, those of a directly density-reachable graph fit the same shape.

For an arbitrarily shaped cluster, if an over-sampling process generates a synthetic instance by locating it along the shortest path in a complete graph, this synthetic instance may be located outside the shape. This effect causes an overlapping problem between the synthetic instance and a negative instance. On the other hand, if a data structure is a directly density-reachable graph, the synthetic instances are located inside the shape because the shortest paths are similar to the skeleton path [33].

A shortest path algorithm applied in a directly density-reachable graph is illustrated in Fig. 4.6. Note that I do not show edges in the graph. The symbols in this figure are described as follows. A circular point represents a positive instance, and the black point represents the centroid of all positive instances in. The point p is one positive instance to be considered. The solid line represents a shortest path in a directly density-reachable graph for the pair of the considered point p and the centroid, whereas

the dotted line represents the shortest path in a complete graph for the same pair. If the shape of a cluster is arbitrarily similar to the cluster in the figure, a synthetic instance generated along the shortest path in a complete graph may overlap with an instance in a majority class located outside of a cluster of a minority class; however, an over-sampling process would not encounter the overlapping problem when using the shortest path from a directly density-reachable graph.

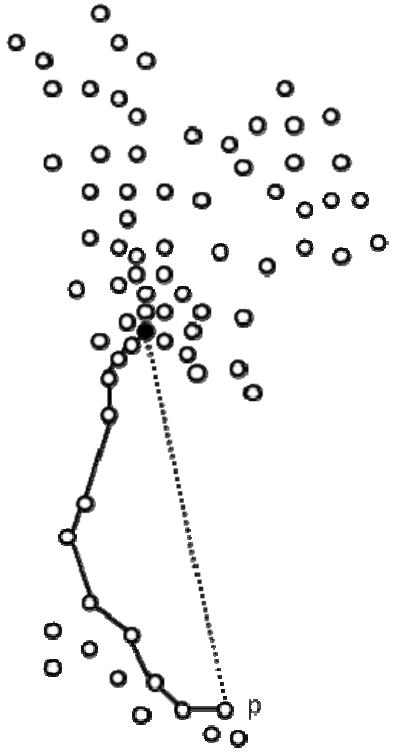

Fig. 4.6. A shortest path found in a directly density-reachable graph.

Normally, the centroid of a cluster is computed by the average of all instances in the cluster. This approach encounters the following problem. If a cluster has some noise instances, which apparently differ from most instances in this cluster, the centroid will be affected by these noise instances. If the noise instances have high values, they will cause the mean value to shift to a larger value while low-valued noise instances will drop from the mean value to a lower number. Fortunately, DBSCAN can discover all clusters and detect noise instances. If some instances are extremely far from most instances,

these instances will be identified as noise instances and excluded from any clusters. Thus, this noise effect cannot interfere with the computation of the centroid. In my research, I use the pseudo-centroid, which is determined by the nearest instance from the centroid, because the centroid may not exist as an actual instance in a cluster.

The flow diagram of the over-sampling framework integrated with DBSMOTE is illustrated in Fig. 4.7 and is described as follows. First, all instances in a minority class D^+ are fed to DBSCAN. Note that a whole dataset D consists of D^+ and D^-, where D^- is all instances in a majority class. Second, DBSCAN produces m clusters: C_1, C_2, …, C_m. These clusters are disjoint sets. In addition, DBSCAN can detect and remove a set of noise instances N. Only the clusters are selected for processing in the next step. Third, each cluster is over-sampled by DBSMOTE. Fourth, DBSMOTE generates m sets of synthetic instances for each cluster: C_1', C_2', …, C_m'. Finally, all sets of synthetic instances are merged with an under-sampling dataset D_{MUTE} operated by MUTE. The result is an over-sampled dataset D'. After the framework finishes, this modified dataset can be fed to any classification algorithm.

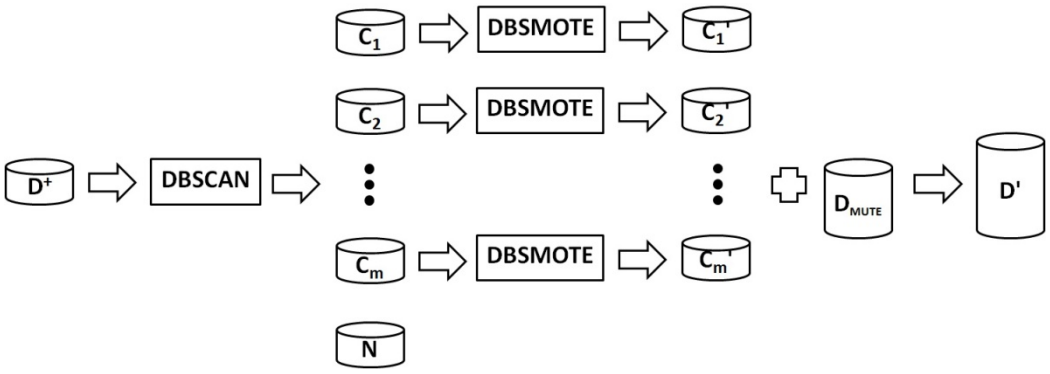

Fig. 4.7. An over-sampling framework integrated with DBSMOTE.

DBSCAN requires the two parameters Eps and $MinPts$, which are passed to all DBSMOTE processes. Moreover, these parameters are global parameters for my over-sampling framework. The values of the parameters can be determined by visualizing a

sorted k-dist graph, as described in Chapter 3.2. This setting guarantees that a directly density-reachable graph is a connected graph; thus, all shortest paths in my graph can be successfully searched. I design to set Eps and $MinPts$ as global parameters because if Eps and $MinPts$ are local parameters for each DBSMOTE process, they will be difficult to determine.

The algorithm DBSMOTE is shown in Fig. 4.8. In this algorithm, the input is a cluster i of instances C_i, Eps ε, and $MinPts$ k while the output is a set i of synthetic instances C_i'. All instances in C_i and C_i' are in a minority class. The values of Eps and $MinPts$ in DBSCAN and DBSMOTE are the same and are derived from the sorted k-dist graph. All variables and all functions in my algorithm are described as follows. In addition, p is a positive instance located in a cluster i of instances C_i, and s is a synthetic instance contained in a set i of synthetic instances C_i'.

Let G be a directly density-reachable graph, which is an output from construct_directly_density-reachable_graph(C_i, ε, k). This function constructs a directly density-reachable graph G from a cluster C_i with respect to Eps ε and $MinPts$ k, relying on Definition 7. These parameters guarantee that a directly density-reachable graph is a connected graph. This function operates in two steps. In the first step, it classifies each instance in a cluster as a core instance or a border instance. The type of all instances is determined because any instances must be directly density-reachable from a core instance. In the second step, for each pair of nodes, if at least one instance in this pair is directly density-reachable from its partner, the edge connecting the pair will be created.

Let c be the pseudo-centroid, which is an output from determine_pseudo-centroid(C_i). This function determines the pseudo-centroid c, which is the nearest instance from the mean of all instances in a cluster C_i. Typically, the centroid of a cluster is computed by the mean of all instances in a cluster, but the centroid may not be an instance in the cluster. Thus, the nearest instance from the mean is selected as the pseudo-centroid.

Let π be a predecessor list, which is an output from Dijkstra(G, c). Dijkstra's algorithm [34] is a graph search algorithm that solves the single-source shortest path problem for a graph whose edge weights are all non-negative. This predecessor list can describe all visited nodes in a shortest path between the pseudo-centroid *c* and any other node. The shortest path is obtained by traversing backwards through the predecessor list. The time complexity of Dijkstra's algorithm is $O(|V|^2 + |E|)$, where $|V|$ is the number of nodes and $|E|$ is the number of edges. This time complexity can be derived from a polynomial function $O(n^2)$ [34] when *n* is equal to $|V|$.

Let § be a shortest path, which is an output from retrieve_shortest_path(π, p, c). This function retrieves a shortest path between a positive instance p and the pseudo-centroid c by traversing π.

Let e be an edge, which is an output from select_random_edge(§). This function randomly selects one edge in the shortest path §.

Let $\{v_1, v_2\}$ be a pair of nodes connected by an edge e, which is an output from get_connected_nodes(e).

For the variables in the loop for, $atti$ is an attribute index, $numattrs$ is the number of attributes, dif is the difference between the numerical values of v_2 and v_1 at the same attribute $atti$, and gap is a random number assigned by generate_random_number(0, 1), which returns a real number in the range from 0 to 1. In addition, $v_1[atti]$, $v_2[atti]$, and $s[atti]$ are the numerical values of the instances at the same attribute $atti$.

To summarize the algorithm, DBSMOTE begins to construct a directly density-reachable graph from a cluster of positive instances with respect to the global parameters Eps and $MinPts$, relying on Definition 7. Next, the pseudo-centroid of the cluster is assigned to the nearest instance from the mean of instances in the cluster. After that, for each instance in the cluster, DBSMOTE generates a synthetic instance along the line segment in the pair of nodes connected by an edge that is randomly selected from a shortest path, embedded in a directly density-reachable graph,

between the instance to the pseudo-centroid. After DBSMOTE is applied to a cluster i, the algorithm returns a set of n_i synthetic instances where n_i is the number of instances in a cluster C_i. Furthermore, the number of synthetic instances generated by my framework depends on the size of a set of noise instances. If a dataset contains t noise instances, the framework will generate (n - t) synthetic instances, where n is the number of instances in a minority class contained in a dataset.

Input: a cluster i of positive instances C_i, Eps ε, and $MinPts$ k
Output: a set i of synthetic instances C_i'
1. $C_i' = \varnothing$
2. G = construct_directly_density-reachable_graph(C_i, ε, k)
3. c = determine_pseudo-centroid(C_i)
4. π = Dijkstra(G, c)
5. for each instance $p \in C_i$ {
6. § = retrieve_shortest_path(π, p, c)
7. e = select_random_edge(§)
8. (v_1, v_2) = get_connected_nodes(e)
9. for ($atti$ = 1 to $numattrs$) {
10. $dif = v_2[atti] - v_1[atti]$
11. gap = generate_random_number(0, 1)
12. $s[atti] = v_1[atti] + gap \cdot dif$
13. }
14. $C_i' = C_i' \cup \{s\}$
15. }
16. return C_i'

Fig. 4.8. DBSMOTE algorithm.

DBSMOTE produces a set of synthetic instances. This set is dense near the pseudo-centroid of a cluster and is sparse far from the pseudo-centroid. In other words, the core information is more emphasized than the border information. The over-sampled dataset causes a classifier to concentrate on learning around the core information spread around the pseudo-centroid more than on in the border information spread near the edge a cluster.

A directly density-reachable graph satisfies the properties that cause similarities between synthetic instances and core instances. In this section, I define two terms, core node and border node. In the graph, a core node is a node that represents a core instance, and a border node represents a border instance. In addition, the weight of an edge is equal to the distance between the pair of connected nodes, and the degree of a node is equal to the number of edges incident to the node. The lemmas related to these properties are described as follows.

Lemma 1 informs us that the weight of each edge in a directly density-reachable graph cannot exceed Eps because Definition 1 and Definition 2 require the distance between a core instance and its Eps-neighbourhood to be no greater than Eps.

In the algorithm DBSMOTE, each synthetic instance is generated along the line of an edge in a directly density-reachable graph. The length of this line is equal to the weight of this edge and is related to the similarity between a synthetic instance and a core instance connected by the edge. If the line is long, the similarity will be low. If the line is short, the similarity will be high. A synthetic instance should be closer to a core instance, which holds the core information, because the similarity between the connected nodes in the pair should not be small. In the worst case, the similarity guarantees that the distance between a synthetic instance and a core instance cannot exceed the distance Eps. This similarity-guaranteed distance Eps is useful for an application with the restricted condition of not allowing an algorithm to create a synthetic instance whose similarity to an instance in a cluster exceeds the specified distance.

In a complete graph in which each node connects to every other node, the maximum weight among all edges incident to a node is equal to the distance from its furthest $(n - 1)^{th}$ nearest neighbour where n is the number of instances in a cluster. In a directly density-reachable graph, that maximum weight is equal to the distance from a node to its furthest k^{th} nearest neighbour. Because k is less than n, the similarity between a core instance and a synthetic instance generated along the line of an edge in a directly density-reachable graph would be more than that in a complete graph.

Lemma 1: The maximum weight of any edge in a directly density-reachable graph is not greater than Eps.

Proof: Definition 1 and Definition 2 restrict the distance between two directly density-reachable instances such that it cannot exceed Eps. □

Lemma 2 informs us that the degree of a core node in a directly density-reachable graph must reach $MinPts$ because a core instance must obtain at least $MinPts$ Eps-neighbourhood relying on the restriction of the core instance condition in Definition 2.

Lemma 3 informs us that the degree of a border node in a directly density-reachable graph must not reach $MinPts$ because a border instance cannot hold the core instance condition in Definition 2.

Because the number of edges incident to a core node is greater than that of a border node, most visited nodes in a shortest path would be core nodes. Thus, synthetic instances would be closer to the core instances than the border instances. DBSMOTE not only emphasizes the generation of synthetic instances near the pseudo-centroid of a cluster over instances near the edge of the cluster, but also locates these instances closer to core instances because the regions around the pseudo-centroid and the core

instances contain significant core information. This phenomenon produces similarity between the information in synthetic instances and the core information.

Lemma 2: The minimum degree of a core node in a directly density-reachable graph is not less than $MinPts$.

Proof: From the core instance condition in Definition 2, a core instance obtains at least $MinPts$ Eps-neighborhood. □

Lemma 3: The maximum degree of a border node in a directly density-reachable graph is less than $MinPts$.

Proof: From the core instance condition in Definition 2, a border instance does not meet this condition. □

By Theorem 1, DBSMOTE takes a polynomial running time $O(n^2)$ when n is the number of instances in a cluster. In the algorithm SMOTE, finding k nearest neighbours takes $O(n)$ and producing a synthetic instance takes $O(1)$ for one instance in a minority class, so the running time of SMOTE is $O(n^2)$. Borderline-SMOTE also takes $O(n^2)$ because the authors did not modify any parts of the algorithm SMOTE; they only changed the input set from all instances to only borderline instances and then fed them to the algorithm SMOTE. Among the SMOTE family, the running time of DBSMOTE is not slower than SMOTE and Borderline-SMOTE. In the over-sampling framework integrated with DBSMOTE, its running time is considered to be the total running time of DBSCAN and DBSMOTE. In the worst case, DBSCAN detects all clusters and a number of noise instances by taking $O(n\lg n)$. Next, DBSMOTE produces a set of synthetic instances by taking $O(n_i^2)$ for a cluster i, where n_i is the number of instances in a cluster i. Because n_i is not greater than n, I can derive that $O(n_i^2)$ is equivalent to $O(n^2)$. After that, merging these sets takes $O(n)$. Because the number of clusters is a constant, the running time of my framework achieves a reasonable time complexity $O(n^2)$.

Theorem 1: (Time complexity)

The time complexity of DBSMOTE is $O(numattrs \cdot n^2)$, where n is the input size.

Proof: All running times of the functions in DBSMOTE are specified as follows.

Each instruction in the lines 1, 10, 12, 14, and 16 takes $O(1)$. Let $T_1(n)$, $T_{10}(n)$, $T_{12}(n)$, $T_{14}(n)$, and $T_{16}(n)$ be the running times of these instructions so there is a positive constant c_i such that $0 \leq T_i(n) \leq c_i$ where $i \in \{1, 10, 12, 14, 16\}$.

The step construct_directly_density-reachable_graph(C_i, ε, k) takes $O(n_2)$ because classifying each instance as a core instance or a border instance takes $O(n_2)$ and then creates all edges in a directly density-reachable graph takes $O(n_2)$. Let $T_2(n)$ be the running time of this function so there are positive constants c_2 and n_0 such that $0 \leq T_2(n) \leq c_2 n_2$ where $\forall n > n_0$.

The step determine_pseudo-centroid(C_i) takes $O(n)$ because computing the mean of all instances in a cluster C_i takes $O(n)$ and then determining the nearest instance from the mean takes $O(n)$. Let $T_3(n)$ be the running time of this function so there are positive constants c_3 and n_0, such that $0 \leq T_3(n) \leq c_3 n$ where $\forall n > n_0$.

The step Dijkstra(G, c) takes $O(n_2)$, relying on the time complexity of Dijkstra's algorithm. Let $T_4(n)$ be the running time of this function so there are positive constants c_4 and n_0 such that $0 \leq T_4(n) \leq c_4 n_2$ where $\forall n > n_0$.

The step retrieve_shortest_path(π, p, c) takes $O(n)$. In the worst case, all nodes in a directly density-reachable graph are visited nodes in a shortest path. Let $T_6(n)$ be the running time of this function so there are positive constants c_6 and n_0 such that $0 \leq T_6(n) \leq c_6 n$ where $\forall n > n_0$.

The step select_random_edge($\S$) takes $O(1)$ because this function randomly selects one edge in a shortest path $\S$. Let $T_7(n)$ be the running time of this function so there is a positive constant c_7 such that $0 \leq T_7(n) \leq c_7$ where $\forall n > n_0$.

The step get_connected_nodes(e) takes $O(1)$ because this function gets the index information for a pair of connected nodes. Let $T_8(n)$ be the running time of this function so there is a positive constant c_8 such that $0 \le T_8(n) \le c_8$ where $\forall n > n_0$.

The step generate_random_number(0, 1) takes $O(1)$ because this function generates a random real number in the range from 0 to 1. Let $T_{11}(n)$ be the running time of this function so there is a positive constant c_{11} such that $0 \le T_{11}(n) \le c_{11}$ where $\forall n > n_0$.

Loop for in line 5 takes n steps.

Loop for in line 9 takes $numattrs$ steps when $numattrs$ is a constant.

Let $T(n)$ be the total running time and be derived as the following inequalities:

$$
\begin{aligned}
0 \le T(n) &\le T_1(n) + T_2(n) + T_3(n) + T_4(n) + n\cdot(T_6(n) + T_7(n) + T_8(n) \\
&\quad + numattrs\cdot(T_{10}(n) + T_{11}(n) + T_{12}(n)) + T_{14}(n)) + T_{16}(n) \\
&\le c_1 + c_2n^2 + c_3n + c_4n^2 \\
&\quad + n\cdot(c_6n + c_7 + c_8 + numattrs\cdot(c_{10} + c_{11} + c_{12}) + c_{14}) + c_{16} \\
&\le (c_1 + c_{16}) + (c_3 + c_7 + c_8 + c_{14} + numattrs\cdot(c_{10} + c_{11} + c_{12}))\cdot n \\
&\quad + (c_2 + c_4 + c_6)\cdot n^2 \\
&\le (c_1 + c_{16})\cdot n^2 + (c_3 + c_7 + c_8 + c_{14} + numattrs\cdot(c_{10} + c_{11} + c_{12}))\cdot n^2 \\
&\quad + (c_2 + c_4 + c_6)\cdot n^2 \\
&\le (c_1 + c_{16} + c_3 + c_7 + c_8 + c_{14} + numattrs\cdot(c_{10} + c_{11} + c_{12}) \\
&\quad + c_2 + c_4 + c_6)\cdot n^2 \\
&\le c\cdot numattrs\cdot n^2
\end{aligned}
$$

Because there is a positive constant c such that $T(n) \le c\cdot numattrs\cdot n^2$ where $\forall n > n_0$, I can derive that $T(n) = O(numattrs\cdot n^2)$ □

By Theorem 2, the shortest paths of each instance to the pseudo-centroid of a cluster exist in a directly density-reachable graph. In addition, the correctness of the algorithm DBSMOTE can be validated by this theorem because the algorithm cannot be halted unless all shortest paths are completely searched by line 6 in the algorithm. If there is even one instance whose shortest path from the pseudo-centroid does not exist, the results will be incorrect, and the algorithm cannot be applied in practice. This theorem is important because it indicates that the algorithm is reliable.

Theorem 2: (Correctness)

There exist all shortest paths between each instance and the pseudo-centroid of a cluster.

Proof: Assume to the contrary that there is an instance z whose shortest path from the pseudo-centroid c of a cluster does not exist. Relying on condition 2 in Definition 5, z and c are in the same cluster; thus, z is density-connected to c. By Definition 4, there must be an instance r such that z and c are density-reachable from r. By Definition 3, there is a chain of instances $p_1, \ldots, p_n, p_1 = r, p_n = z$ such that p_{i+1} is directly density-reachable from p_i, where n is the number of instances in this chain and i is an integer in the range from 1 to n - 1. By Definition 7, p_{i+1} is directly density-reachable from p_i; thus, an edge connecting p_{i+1} and p_i exists. This sequence of instances is a path between r and z. Because z and c are density-reachable from r, a path between z and c also exists, contradicting my assumption that there is a path between z and c. Thus, all shortest paths between each instance and the pseudo-centroid in the cluster exist when the instances in this pair are represented as nodes in a directly density-reachable graph. □

CHAPTER V

Experiment

In this chapter, I describe my collected datasets form UCI and show my experimental results with their statistical tests as follows.

In my experiment, I compared three performance measures: accuracy, F-value, and AUC of my framework with those of SMOTE and Safe-Level-SMOTE, by applying three classifiers: a decision tree C4.5, a rule-based classifier RIPPER, and a neural network model MLP, on five UCI imbalanced datasets: Glass, Letter Recognition, Page-Blocks, Satimage, and Segmentation, with multiple minority classes. In addition, I run the paired t-tests to test a difference in means across the paired observations of my framework with SMOTE and Safe-Level-SMOTE.

1. Dataset

I selected the multi-class datasets with various degrees of imbalance from the UCI Repository of Machine Learning Databases [35], Glass Identification, Letter Recognition, Page-Blocks, Satimage, and Image Segmentation, which are shown in Table 5.1.

I randomly split each dataset except Satimage, which has separated training and test sets, into a training set (2/3) and a test set (1/3). In addition, target classes were selected as minority classes and remaining classes were merged as a majority class. In order to avoid the randomness of SMOTE techniques, I determined median accuracy, F-value, and AUC from 3 independent running.

Table 5.1. The descriptions of UCI datasets in the experiments.

Dataset	Attributes	Instances	Class ID	Minority Class Name	Minority Class %
Glass	10	17	3	Vehicle Windows	7.94
		13	5	Containers	6.07
Letter Recognition	16	736	3	C	3.68
		734	8	H	3.67
		734	26	Z	3.67
Page-blocks	10	329	2	Horizontal Line	6.01
		88	4	Vertical Line	1.61
		115	5	Picture	2.10
Satimage	36	479	2	Cotton Crop	10.80
		415	4	Damp Grey Soil	9.36
		470	5	Soil with Vegetation Stubble	10.60
Segmentation	19	330	3	Foliage	14.29
		330	5	Window	14.29

2. Experimental Result

For my experimental design, I used the performance measures accuracy, F-value (as Precision and Recall), and AUC to evaluate the over-sampling techniques SMOTE, Safe-Level-SMOTE (SAFE), and DBSMOTE (DBS). The value of β in F-value was set to 1. The number of k nearest neighbour to be over-sampled was set to 5 as the default value of SMOTE [8]. The standard classifiers, Decision Tree C4.5, RIPPER, and Multilayer Perceptron (MLP), were applied. The experimental results from all datasets are illustrated in Tables 5.2 through 5.5.

Table 5.2. Accuracy results when applying SMOTE family on UCI datasets.

Classifier	Dataset	DBS	SMOTE	SAFE
C4.5	Glass	**84.21**	76.31	78.94
	Letter Recognition	96.56	**97.01**	96.57
	Page-blocks	**97.85**	97.30	97.63
	Satimage	85.85	86.30	**86.95**
	Segmentation	**96.73**	95.69	96.08
Ripper	Glass	**84.21**	78.94	80.26
	Letter Recognition	96.84	**97.01**	96.69
	Page-blocks	97.46	97.52	**97.57**
	Satimage	**88.25**	86.80	87.15
	Segmentation	**95.30**	**95.30**	94.90
MLP	Glass	**90.78**	81.57	86.84
	Letter Recognition	**96.35**	95.94	95.97
	Page-blocks	**96.97**	96.53	96.47
	Satimage	**90.20**	88.75	88.95
	Segmentation	**97.65**	97.25	96.86

Table 5.3. F-value results when applying SMOTE family on UCI datasets.

Classifier	Dataset	Class	DBS	SMOTE	SAFE
C4.5	Glass	3	0.286	**0.400**	0.167
		5	**0.750**	0.333	0.727
	Letter Recognition	3	**0.888**	0.883	0.883
		8	0.742	**0.798**	0.754
		26	**0.905**	0.899	0.892
	Page-blocks	2	**0.936**	0.933	0.933
		4	**0.867**	**0.867**	**0.867**
		5	**0.725**	0.676	0.712
	Satimage	2	**0.958**	0.945	0.935
		4	**0.603**	0.564	0.580
		5	0.813	0.814	**0.832**
	Segmentation	3	**0.939**	0.930	0.933
		5	**0.905**	0.867	0.890
Ripper	Glass	3	**0.400**	0.375	0.375
		5	**0.667**	0.600	0.444
	Letter Recognition	3	**0.915**	0.901	0.899
		8	0.733	**0.743**	0.720
		26	**0.935**	0.933	0.926
	Page-blocks	2	0.916	0.919	**0.926**
		4	**0.906**	0.867	0.867
		5	0.727	0.714	**0.740**
	Satimage	2	**0.951**	0.944	0.935
		4	**0.624**	0.607	0.612
		5	0.833	0.828	**0.839**
	Segmentation	3	**0.932**	0.920	0.912
		5	0.863	**0.865**	0.851

Table 5.3. F-value results when applying SMOTE family on UCI datasets (Continue).

Classifier	Dataset	Class	DBS	SMOTE	SAFE
MLP	Glass	3	**0.600**	0.167	0.200
		5	**0.833**	0.769	**0.833**
	Letter Recognition	3	**0.865**	0.852	0.848
		8	**0.777**	0.728	0.735
		26	**0.881**	0.861	0.861
	Page-blocks	2	**0.903**	0.870	0.867
		4	**0.800**	**0.800**	0.792
		5	**0.718**	0.684	0.684
	Satimage	2	**0.971**	0.951	0.969
		4	**0.679**	0.662	0.647
		5	**0.880**	0.859	0.861
	Segmentation	3	**0.960**	0.959	0.951
		5	**0.929**	0.915	0.906

Table 5.4. AUC results when applying SMOTE family on UCI datasets.

Classifier	Dataset	Class	DBS	SMOTE	SAFE
C4.5	Glass	3	**0.783**	0.740	0.669
		5	**0.907**	0.747	0.870
	Letter Recognition	3	**0.978**	0.964	0.958
		8	0.930	**0.951**	0.942
		26	0.958	**0.961**	**0.961**
	Page-blocks	2	0.971	**0.972**	**0.972**
		4	**0.946**	**0.946**	0.929
		5	0.852	0.890	**0.895**
	Satimage	2	**0.985**	0.973	0.977
		4	**0.852**	0.795	0.787
		5	**0.927**	0.892	0.908
	Segmentation	3	**0.981**	0.980	0.977
		5	**0.961**	0.917	0.922
Ripper	Glass	3	**0.714**	0.706	0.713
		5	**0.786**	0.714	0.695
	Letter Recognition	3	**0.966**	0.961	0.959
		8	0.899	**0.902**	0.873
		26	**0.977**	0.974	0.975
	Page-blocks	2	0.974	**0.981**	0.975
		4	**0.981**	0.933	0.934
		5	**0.897**	0.825	0.826
	Satimage	2	**0.980**	0.968	0.965
		4	**0.858**	0.844	0.836
		5	0.918	**0.923**	0.905
	Segmentation	3	**0.987**	0.981	0.947
		5	**0.951**	0.927	0.921

Table 5.4. AUC results when applying SMOTE family on UCI datasets (Continue).

Classifier	Dataset	Class	DBS	SMOTE	SAFE
MLP	Glass	3	**0.867**	0.838	0.840
		5	**0.990**	0.983	0.861
	Letter Recognition	3	**0.995**	0.977	0.960
		8	**0.961**	0.951	0.956
		26	**0.989**	0.985	0.981
	Page-blocks	2	**0.983**	0.978	0.979
		4	**0.992**	0.981	0.990
		5	0.954	**0.955**	0.954
	Satimage	2	**0.999**	0.996	0.998
		4	**0.941**	0.939	0.936
		5	**0.989**	0.972	0.976
	Segmentation	3	**0.998**	**0.998**	0.997
		5	0.994	**0.995**	0.991

For the statistical analysis illustrated in Tables 5.5 through 5.7, I applied the paired t-tests to all the accuracy, F-value and AUC results above. For each test, the null and alternative hypotheses were

H_0: $\mu_1 - \mu_2 = 0$
H_1: $\mu_1 - \mu_2 \neq 0$

where μ_1 is the mean of DBSMOTE and μ_2 is the mean of SMOTE or Safe-Level-SMOTE. All pairs of variances were not significantly different. For each result, I did two paired t-tests: DBSMOTE to SMOTE and DBSMOTE to Safe-Level-SMOTE. The significance level (α) was set to 0.05. If $P(T \leq t)$ two-tail < α, H_0 is rejected, which meant that there was a difference in means across the paired observations.

Table 5.5. t-test: paired two sample for means on accuracy.

	DBS	SMOTE	DBS	SAFE
Mean	93.014	91.21467	93.014	91.85533
Variance	27.11577	55.94568	27.11577	41.6465
Observations	15	15	15	15
Pearson Correlation	0.944343		0.975887	
Hypothesized Mean Difference	0		0	
Degree of Freedom	14		14	
t Stat	2.261047		2.519068	
P(T $\leq$ t) one-tailed	0.020104		0.012272	
t Critical one-tailed	1.76131		1.76131	
P(T $\leq$ t) two-tailed	**0.040207**		**0.024545**	
t Critical two-tailed	2.144787		2.144787	

Table 5.6. t-test: paired two sample for means on F-value.

	DBS	SMOTE	DBS	SAFE
Mean	0.808077	0.77441	0.808077	0.777051
Variance	0.023457	0.036204	0.023457	0.037869
Observations	39	39	39	39
Pearson Correlation	0.862332		0.940544	
Hypothesized Mean Difference	0		0	
Degree of Freedom	38		38	
t Stat	2.168379		2.67111	
P(T $\leq$ t) one-tailed	0.018229		0.005534	
t Critical one-tailed	1.685954		1.685954	
P(T $\leq$ t) two-tailed	**0.036458**		**0.011068**	
t Critical two-tailed	2.024394		2.024394	

Table 5.7. t-test: paired two sample for means on AUC.

	DBS	SMOTE	DBS	SAFE
Mean	0.937718	0.920897	0.937718	0.915641
Variance	0.004557	0.006839	0.004557	0.006971
Observations	39	39	39	39
Pearson Correlation	0.924886		0.922806	
Hypothesized Mean Difference	0		0	
Degree of Freedom	38		38	
t Stat	3.211854		4.108829	
P(T $\leq$ t) one-tailed	0.001343		0.000102	
t Critical one-tailed	1.685954		1.685954	
P(T $\leq$ t) two-tailed	0.002685		0.000204	
t Critical two-tailed	2.024394		2.024394	

To summarize the experimental results, it is apparent that DBSMOTE achieves the best accuracy, F-value (as Precision and Recall), and AUC when applying various types of classifiers. According to the paired t-tests, the performances of DBSMOTE are significantly better than that of SMOTE and Safe-Level-SMOTE for all measures.

CHAPTER VI

Discussion and Conclusion

In the last chapter, I discuss and summarize my research and also point out my future works as follows.

The trade-off between re-sampling techniques in this thesis is revealed as following facts. SMOTE consumes the least execution time in seconds but fails to operate on an overlapping region. Safe-Level-SMOTE can treat the overlapping problem but does not apply the features scaling process. DBSMOTE efficiently remedy the overlapping problem but consumes the least execution time in seconds. It has been evidenced that SMOTE is the fastest in the experiment; however in the class imbalanced problem I aim to archive the predictive performance but not the execution time so DBSMOTE is the best technique for handling this kind of problem.

The trade-off between re-sampling techniques in this thesis is revealed as following facts. SMOTE consumes the least execution time in seconds but fails to operate on an overlapping region. Safe-Level-SMOTE can treat the overlapping problem but does not apply the features scaling process. DBSMOTE efficiently remedy the overlapping problem but consumes the least execution time in seconds. It has been evidenced that SMOTE is the fastest in the experiment; however in the class imbalanced problem I aim to archive the predictive performance but not the execution time so DBSMOTE is the best technique for handling this kind of problem.

In conclusion, the class imbalanced problem has got more attentions among machine learning society. Unfortunately, traditional data mining techniques are still unsatisfactory when applications encounter this kind of problem. In this thesis, I propose the density-based over-sampling framework DBSMOTE which concentrates to operate on the core of a cluster of a minority class rather than the border of this cluster because this core contains significant information; as a result, a classifier emphasizes to learn on this core. Moreover, I also provide future works to improve both predictive performance and execution time of my framework.

1. Discussion

SMOTE family is the group of over-sampling techniques based on SMOTE and consists of an original SMOTE, Safe-Level-SMOTE and DBSMOTE. The discussion among these techniques is described as Table 6.1.

For SMOTE, the main advantage is that the execution time of SMOTE in the experiment is the fastest comparing with the other techniques in SMOTE family because computing k nearest neighbours consumes the low cost; in addition, the disadvantage is that SMOTE cannot treat the overlapping regions because the synthetic instances generated in these regions would crash into the negative instances.

For Safe-Level-SMOTE, the advantage is that Safe-Level-SMOTE can treat the overlapping regions because the synthetic instances generated in these regions would be closer to the core of a cluster rather than the border of this cluster so these synthetic instances would not crash into the negative instances; in addition, the disadvantage is that an instance which its attributes have highly values might not be considered as the k nearest neighbours because the features scaling approach is not applied.

For DBSMOTE, the advantage is that DBSMOTE can efficiently treat the overlapping regions because the synthetic instances generated in these regions located closer to the safe regions which contain the core information of a minority class so a classifier is induced conveniently to separate the regions between a minority class and a majority class; in addition, the disadvantage is that the execution time of DBSMOTE in the experiment is the slowest comparing with the other techniques in SMOTE family because Dijkstra's algorithm consumes the high cost.

Table 6.1. The Discussion on SMOTE family.

Technique	Concept	Pros	Cons
SMOTE	• SMOTE over-samples all instances spreading throughout a cluster in every region. • SMOTE generates all synthetic instances along a line joining each positive instance and its randomly selected positive nearest neighbours. • The density of the synthetic instances is not dense in any particular regions.	• SMOTE consumes the least execution time in the experiment comparing with the other techniques. • SMOTE is widely available in data mining softwares such as WEKA. • SMOTE is easy to implement in various computer languages.	• SMOTE poorly handles the overlapping regions.

Table 6.1. The Discussion on SMOTE family (Continue).

Technique	Concept	Pros	Cons
Safe-Level-SMOTE	• Each positive instance has its safe level computed by the number of positive instances among the k nearest neighbours. • Each synthetic instance is generated along the same over-sampling line of SMOTE but positioning closer to a larger safe level instance. • The density of the synthetic instances is similar to that of SMOTE but not spreading in the overlapping regions.	• Safe-Level-SMOTE handles the overlapping regions.	• An instance with a highly value-feature may not be considered as the k nearest neighbours.

Table 6.1. The Discussion on SMOTE family (Continue).

Technique	Concept	Pros	Cons
DBSMOTE	• Each synthetic instance is generated along the shortest path from each positive instance to the pseudo-centroid of a minority class-cluster when these instances in the pair are represented as nodes in a directly density-reachable graph. • The density of the synthetic instances is dense nearby the pseudo-centroid and is sparse far from the pseudo-centroid.	• DBSMOTE efficiently handles the overlapping regions.	• Although the time complexity is $O(n^2)$, the execution time in the experiment is slower than the other techniques.

2. Conclusion

Class imbalanced problems are beginning to receive more attention among data miners and researchers. An application encounters this problem when the classes in a dataset are imbalanced. In addition, there are many techniques for handling the problems, such as over-sampling and under-sampling. Unfortunately, traditional data mining techniques are not capable of solving this kind of problem. I have designed an efficient technique called DBSMOTE based on the density concept for dealing with class imbalanced problems.

DBSMOTE begins to run DBSCAN to discover arbitrarily shaped clusters and to detect noise instances to be deleted and then execute SMOTE to generate synthetic instances inside the shapes of these clusters. These instances tend to avoid appearing in any regions of a majority class so the prediction rate on a minority class would be improved.

According to the experimental results, the performance of DBSMOTE evaluated by F-value (as Precision and Recall), and AUC is better than that of SMOTE and Safe-Level-SMOTE when applying various types of classifiers. This phenomenon stems from DBSMOTE's generation of all synthetic instances along the shortest paths between each instance and the pseudo-centroid of a cluster in a directly density-reachable graph. Thus, the synthetic instances are dense near the pseudo-controid and sparse far from the pseudo-controid. Consequently, these synthetic instances cause a classifier to concentrate on the core information located around the pseudo-centroid instead of on the border information located around the edge of a cluster. In summary, the synthetic instances can improve prediction performance on a minority class. Furthermore, the statistical analysis supports my conclusions.

Analysis of the algorithm reveals that DBSMOTE takes $O(numattr \cdot n^2)$ when n is the size of an input set, and both SMOTE and Safe-Level-SMOTE take the same running time. The correctness of DBSMOTE is also validated. The information contained in the synthetic instances is more similar to the core information than the border information.

3. Future Work

Although the experimental results have provided evidence that DBSMOTE can successfully classify an imbalanced dataset, there is considerable room for future work in this line of research. First, different density-based clustering algorithms could replace DBSCAN by integrating with DBSMOTE. Second, pruning a directly density-reachable graph might improve the performance of DBSMOTE. Third, automatic determination of the number of synthetic instances generated by DBSMOTE and the values of Eps and $MinPts$ should be addressed.

References

[1] Chawla, N.V., Japkowicz, N., Kolcz, A.: Editorial: Special Issue on Learning from Imbalanced Data Sets. SIGKDD Explorations, vol. 6, no. 1, pp. 1--6 (2004)

[2] Japkowicz, N.: Class imbalance: Are we focusing on the right issue?. Proceedings of the 20th International Conference on Machine Learning, pp. 17--23. Washington, District of Columbia, USA (2003)

[3] Japkowicz, N.: The Class Imbalance Problem: Significance and Strategies. Proceedings of the 2000 International Conference on Artificial Intelligence, pp. 111--117. Las Vegas, Nevada, USA (2000)

[4] Jo, T., Japkowicz, N.: Class Imbalances versus Small Disjuncts. SIGKDD Explorations, vol. 6, no. 1, pp. 40--49 (2004)

[5] Bunkhumpornpat, C., Sinapiromsaran, K., Lursinsap, C.: DBSMOTE: Density-Based Synthetic Minority Over-sampling Technique. Applied Intelligence, DOI: 10.1007/s10489-011-0287-y (2011)

[6] Bunkhumpornpat, C., Sinapiromsaran, K., Lursinsap, C.: MUTE: Majority Under-sampling TEchnique. Proceedings of the 8th International Conference on Information, Communications, and Signal Processing. Singapore (2011)

[7] Bunkhumpornpat, C., Sinapiromsaran, K., Lursinsap, C.: Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-sampling TEchnique for handling the class imbalanced problem. Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Bangkok, Thailand. Lecture Notes in Artificial Intelligence, vol. 5476, pp. 475--482. Springer, Heidelberg (2009)

[8] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic Minority Over-Sampling TEchnique. Journal of Artificial Intelligence Research, vol. 16, pp. 341--378 (2002)

[9] Han, H., Wang, W.-Y., Mao, B.-H.: Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) the 2005 International Conference on Intelligent Computing, Hefei, China. Lecture Notes in Computer Science, vol. 3644, pp. 878--887.

Springer, Heidelberg (2005)

[10] Kubat, M., Holte, R.C., Matwin, S.: Learning When Negative Examples Abound. Proceedings of the 9th European Conference on Machine Learning, pp. 146--153. Prague, Czech Republic (1997)

[11] Ezawa, K.J., Singh, M., Norton, S.W.: Learning Goal Oriented Bayesian Networks for Telecommunications Risk Management. Proceedings of the 13th International Conference on Machine Learning, pp. 139--147. Bari, Italy (1996)

[12] Fan, W., Miller, M., Stolfo, S., Lee, W., Chan, P.K.: Using Artificial Anomalies to Detect Unknown and Known Network Intrusions. Proceedings of the 1st IEEE International Conference on Data Mining, pp. 123--130. San Jose, California, USA (2001)

[13] Fawcett, T., Provost, F.: Combining Data Mining and Machine Learning for Effective User Profile. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, pp. 8--13. Portland, Oregon, USA (1996)

[14] Japkowicz, N., Myers, C., Gluck, M.: A Novelty Detection Approach to Classification. Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp. 518--523. Montreal, Canada (1995)

[15] Kubat, M., Holte, R.C., Matwin, S.: Machine Learning for the Detection of Oil Spills in Satellite Radar Images. Machine Learning, vol. 30, no. 2-3, pp. 195--215 (1998)

[16] Ling, C.X., Li, C.: Data Mining for Direct Marketing Problems and Solutions. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, pp. 73--79. New York, New York City, USA (1998)

[17] Woods, K.S., Doss, C.C., Bowyer, K.W., Solka, J.L., Priebe, C.E., Kegelmeyer, W.P.: Comparative Evaluation of Pattern Recognition Techniques for Detection of Microcalcifications in Mammography. International Journal of Pattern Recognition and Artificial Intelligence, vol. 7, no. 6, pp. 1417--1436 (1993)

[18] Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. SIGKDD Explorations, vol. 6, no. 1, pp. 20--29 (2004)

[19] Kubat, M., Matwin, S.: Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. Proceedings of the 14th International Conference on Machine Learning, pp. 179—186. Nashville, Tennessee, USA (1997)

[20] Tetko, I.V., Livingstone, D.J., Luik, A.I.: Neural Network Studies. 1. Comparison of Overfitting and Overtraining. Journal of Chemical Information and Computer Sciences, vol. 35, no. 5, pp. 826--833 (1995)

[21] Domingos, P.: Metacost: A General Method for Making Classifiers Cost-sensitive. Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 155--164. San Diego, California, USA (1999)

[22] Fan, W., Stolfo, S.J., Zhang, J., Chan, P.K.: AdaCost: misclassification cost-sensitive boosting. Proceedings of the 16th International Conference on Machine Learning, pp. 97--105. Bled, Slovenia (1999)

[23] Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., Brunk, C.: Reducing Misclassification Costs. Proceedings of the 11th International Conference on Machine Learning, pp. 217--225. San Francisco, California, USA (1994)

[24] Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: SMOTEBoost: Improving prediction of the Minority Class in Boosting. Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 107--119. Cavtat-Dubrovnik, Croatia (2003)

[25] Freund, Y., Schapire, R.: Experiments with a New Boosting Algorithm. Proceedings of the 13th International Conference on Machine Learning, pp. 325--332. Bari, Italy (1996)

[26] Lewis, D.D., Catlett, J.: Heterogeneous Uncertainty Sampling for Supervised Learning. Proceedings of the 11th International Conference on Machine Learning, pp. 148--156. New Brunswick, New Jersey, USA (1994)

[27] Kamber, M., Han, J.: Data mining: Concepts and Techniques, 2nd Edition. Morgan Kaufman, USA (2000)

[28] Bradley, A.P.: The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms. Pattern Recognition, vol. 30, no. 6, pp. 1145--1159 (1997)

[29] Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, USA

(1992)

[30] Cohen, W. W.: Fast Effective Rule Induction. Proceedings of 12th International Conference on Machine Learning, pp. 115--123. Lake Tahoe, California, USA (1995)

[31] Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226--231. Portland, Oregon, USA (1996)

[32] Jungnickel, D.: Graphs, Networks and Algorithms. Springer, Heidelberg (2003)

[33] Bai, X., Yang, X., Yu, D., Latecki, L.J.: SKELETON-BASED SHAPE CLASSIFICATION USING PATH SIMILARITY. International Journal on Pattern Recognition and Artificial Intelligence, vol. 22, no. 4, pp. 733--746 (2008)

[34] Corman, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd Edition. The MIT Press, USA (2001)

[35] Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases. http://archive.ics.uci.edu/ml/. Department of Information and Computer Sciences, University of California, Irvine, California, USA (2010)

Biography

I received my B.Eng. in Computer Engineering from Rangsit University and my M.S. in Computer Science from Chiang Mai University. I am a lecturer in the Department of Computer Science, Chiang Mai University. I am studying a Ph.D. Program in Computer Science at Chulalongkorn University. My research focus is Data Mining- especially in the Class Imbalanced Problems and Clustering.

Chumphol Bunkhumpornpat