THE REVIEW ALGORITHM OF SMOTE FAMILY

Jatuporn Rueangkhetpit

Saranya Pagorntadapan

Inprapa Aunsanee

An Independent Study Submitted in Partial Fulfillment

of the Requirements for the Bachelor of Science Degree

in Mathematics

April 2018

Advisor and Dean of School of Science have considered the independent study entitled "The Review Algorithm of Smote Family" submitted in partial fulfillment of the requirements for the Degree of Bachelor of Science Program in Mathematics is hereby approved

(Assist. Prof. Dr. Kamonrat Nammanee)

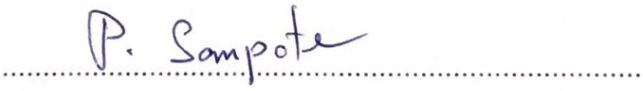
Chairman

(Mr. Suebkul Kanchanasuk)

Advisor

(Ms. Nilobol Kamyun)

Committee



(Assoc. Prof. Preeyanan Sanpote)

Dean of School of Science

April 2018

Acknowledgement

First, we would like to thanks our advisor, Mr. Suebkul Kanchanasuk for his kindness, invaluable advices untill he help and encourage our research.

In addition, we would like to thank Assist. Prof .Dr. Kamonrat Nammanee and Ms. Nilobol Kamyun, who give advice and suggestion for this independent study.

Finally, we would like to thank our parents and friends for their help and encouragement.

Jatuporn Rueangkhetpit

Saranya Pagorntadapan

Inprapa Aunsanee

ชื่อเรื่อง การสำรวจความคิดเห็นของครอบครัว SMOTE

ผู้ศึกษาค้นคว้า นางสาวจตุพร เรืองเขตพิศ
นางสาวศรัญญา ปกรณ์ธาดาพันธ์
นางสาวอินประภา อุ่นเสนีย์
อาจารย์ที่ปรึกษา อาจารย์สืบกุล กาญจนสุกร์

วิทยาศาสตรบัณฑิต สาขาวิชาคณิตศาสตร์
คำสำคัญ ปัญหาการจำแนกประเภท, ปัญหาความไม่สมดุลของคลาส, เทคนิคการสุ่มตัวอย่างเพิ่ม, การจัดกลุ่มตามความหนาแน่น

บทคัดย่อ

ปัญหาการจำแนกประเภทเป็นหัวข้อหนึ่งในการทำเหมืองข้อมูลเพื่อทำนายกลุ่มของข้อมูลหรือเป้าหมายของข้อมูลที่จะมาถึง ให้อยู่ในกลุ่มของข้อมูลที่ถูกต้อง ข้อมูลที่มีอยู่แล้วในอดีตจะถูกนำมาใช้เพื่อสร้างตัวแบ่งประเภท ปัญหานี้เรียกว่าปัญหาความไม่สมดุล เมื่อจำนวนข้อมูลในกลุ่มหนึ่งน้อยกว่ากลุ่มอื่นและแสดงว่าเป็นกลุ่มซึ่งมีสมาชิกน้อย แม้จะมีค่าความถูกต้องสูงของตัวจำแนกประเภทที่รู้จักกันดีสำหรับปัญหาความไม่สมดุลนั้น แต่กลุ่มน้อยส่วนใหญ่ยังไม่ได้รับการจำแนกประเภท เพื่อลดอัตราการผิดพลาดของการทำนายผิดให้ข้อมูลในกลุ่มน้อย ควรได้รับการพิจารณาจากขั้นตอนวิธีการจัดหมวดหมู่ เทคนิคที่ได้รับความนิยมคือ เทคนิคการสุ่มตัวอย่างเพิ่ม ที่เรียกว่า SMOTE (Synthetic minority over–sampling technique) ในการศึกษาครั้งนี้ เราศึกษาแนวคิดนี้ซึ่งดึงเทคนิคของ SMOTE family algorithm ได้แก่ SMOTE, Borderline–SMOTE และ Safe–Level SMOTE

Title	The Review Algorithm of Smote Family
Author	Jatuporn Rueangkhetpit
	Saranya Pagorntadapan
	Inprapa Aunsanee
Advisor	Mr. Suebkul Kanchanasuk
Bachelor of Science	Program in Mathematics
Keywords	Classification Problem, Class imbalance Problem,
	Over-sampling Technique, Density-based Clustering

Abstract

Classification problem is one of the topics in data mining to predict the class or target of the upcoming data into the correct class. The historical known class data is used to build the classifier. The problem is called imbalance problem when the number of data in one class is less than the other class and denoted the minority class as the class which contains the fewer members. Despite the high accuracy value of the well-known classifiers for the imbalance problem, the most misclassified class of minority class still occurred. To reduce the false positive rate, the data in minority class should be attended from the classification algorithm. The one favorite technique is over-sampling technique, called SMOTE (Synthetic minority over-sampling technique).In this study, we investigate the concept, extract the technique of the family of SMOTE algorithm, as well as, SMOTE, Borderline-SMOTE and Safe-Level SMOTE.

Contents

List of Figures

Chapter 1

Introduction

1.1 The importance and the source of the research problem

Data mining is the process performing with large amounts of data to discover The patterns and relationships which is hidden in the data set. In current, Data mining has been applied in many types of work such as Business, Science and Medical [6]. One technique for data mining that our group used in seminars that is, Classification which assortment of data by finding the master series, or a series of works that describe and classify information. The objective is to predict the type of material or information that does not identify the type of data that model generated from the analysis of Training Data. It may be a data group that identifies a category or group. The format of the show has several such as Classification Rules, Decision Trees and cost sensitive learning etc [6]. Our seminars of the group is the study of the student's drop-out prediction using data and social behaviour of the students. By using the techniques

cost sensitive learning obtain that students who drop-out of students that success is very different. Also known as Imbalance data that the data set contains a number of data in each class are different. When the data is very different, there is a problem of unbalanced classification [3]. In this classification, interested only in most data class without considering the minority class. Without at least some cases, the information in this class may have great importance should not be overlooked. For example, from our seminar. The imbalanced classification problem is that the classifier will classify the student as a success, without considering drop-out students, which should not be overlooked. When these problems occur, there are several approaches that can be used to solve imbalanced classification problems such as random under-sampling, random over-sampling, clustering-based over sampling, and additional sampling technique called SMOTE [11]. In this independent study, our group chose SMOTE family techniques, SMOTE, Borderline-SMOTE, and Safe-Level SMOTE to solve imbalanced classification problems. This will only interest the data in the minority class. There is a way to increase the number of data types with less data, increasing the amount of data close to the most available ones. Randomly increment a value and find the distance between the values selected for each value. Choose the closest value. This will result in better classification of information [10].

1.2 Objective

1. To study the specific classification problems for class imbalance data.

2. To study the over-sampling techniques, called SMOTE family algorithm, such as, SMOTE, Borderline-SMOTE and Safe-Level SMOTE.

3. To describe the advantage and disadvantage of the SMOTE family algorithm.

Chapter 2

Literature Review and Related Studies

Research of SMOTE, Borderline-SMOTE and Safe-level-SMOTE. The researcher studied concept, theories and related documents as a guideline for the following research.

2.1 Classification Problems

Classification is one of data mining techniques that have worked on classification data. for to predict the data we are interested to study by using classifier on classification. In this classification, we use the data 2 class and the data contained in the class is different, that is, there is a lot of data in one class and one another class there is a little of data. To which will use the accuracy value decition, the result of the classification. In this classification, Most are classified only in majority class and minority class are not classified, But in some case minority class maybe important or

need to classification. This problem is said to be classification problem.

2.1.1 Class Imbalanced Problems

In case of binary classification problem which the numbers of member in each class is varied often lead to unsatisfactory results from the focus on the accuracy result without concern the prediction of new observations, especially for the small class. In this context, imbalanced classes simply means that the number of observations of one class (usu. positive or majority class) by far exceeds the number of observations of the other class (usu. negative or minority class). This setting can be observed fairly often in practice and in various disciplines like credit scoring, fraud detection, medical diagnostics or churn management [8].

Most classification methods work best when the number of observations per class are roughly equal. The problem with imbalanced classes is that because of the dominance of the majority class classifiers tend to ignore cases of the minority class as noise and therefore predict the majority class far more often. In order to lay more weight on the cases of the minority class, there are numerous correction methods which tackle the imbalanced classification problem [8].

2.2 Over-sampling Technique

2.2.1 Introduction

Oversampling in data analysis are techniques used to adjust the class distribution of a data set (i.e. the ratio between the different classes/categories represented) [12].

The usual reason for oversampling is to correct for a bias in the original dataset.

One scenario where it is useful is when training a classifier using labeled training data from a biased source, since labeled training data is valuable but often comes from un-representative sources [12].

For example, suppose we have a sample of 1000 people of which 66.7% are male. We know the general population is 50% female, and we may wish to adjust our dataset to represent this. Simple oversampling will select each female example twice, and this copying will produce a balanced dataset of 1333 samples with 50% female. Simple under-sampling will drop some of the male samples at random to give a balanced dataset of 667 samples, again with 50% female [12].

There are also more complex oversampling techniques, including the creation of artificial data points [12].

2.3 SMOTE family algorithm

2.3.1 SMOTE

Introduction of SMOTE

There are a number of methods available to over-sample a dataset used in a typical classification problem (using a classification algorithm to classify a set of images, given a labeled training set of images). The most common technique is known as SMOTE: Synthetic Minority Over-sampling Technique. To illustrate how this technique works consider some training data which has s samples, and f features in the feature space of the data. Note that these features, for simplicity, are continuous. As an example, consider a dataset of birds for clarification. The feature space for the minority class for which we want to over-sample could be beak length, wingspan, and weight (all

continuous). To then over-sample, take a sample from the dataset, and consider its k nearest neighbors (in feature space). To create a synthetic data point, take the vector between one of those k neighbors, and the current data point. Multiply this vector by a random number x which lies between 0, and 1. Add this to the current data point to create the new, synthetic data point[13].

Algorithm SMOTE

Algorithm *SMOTE*(T, N, k)

Input: Number of minority class samples ; T ; Amount of SMOTE $N\%$;

Number of nearest neighbors k

Output: $(N/100) * T$ synthetic minority class samples

1. (∗ *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* ∗)
2. **if** $N < 100$
3. **then** Randomize the T minority class samples
4. $T = (N/100) * T$
5. $N = 100$
6. **endif**
7. $N = (\text{int})(N/100)$(∗ *The amount of SMOTE is assumed to be in integral multiples of 100.* ∗)
8. k = Number of nearest neighbors
9. *numattrs* = Number of attributes
10. *Sample*[ ][ ]: array for original minority class samples
11. *newindex*: keeps a count of number of synthetic samples generated, initializd to 0
12. *Synthetic*[ ][ ]: array for synthetic samples

(∗ *Compute k nearest neighbors for each minority class sample only.* ∗)

13. **for** $i \leftarrow 1$ **to** T
14. Compute *k nearest neighbors for i,and save the indices in the nnarray*
15. Populate(*N , i , nnarray*)
16. **endfor**

Populate(N , i , *nnarray*) ($*$ *Function to ganerate the synthetic samples.* $*$)

-p0 17. **while** $N \neq 0$

18. Choose a random number between 1 and k, call it *nn*. This step chooses one of the k nearest neighbors of i.

19. **for** $attr \leftarrow 1$ **to** *numattrs*

20. Compute: $dif = Sample[nnarray[nn]][attr]$ - $Sample[i][attr]$

21. Compute: gap = random number between 0 and 1

22. $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$

23. **endfor**

24. $newindex{+}{+}$

25. $N = N$ - 1

26. **endwhile**

27. **return** ($*$ *End of Populate.* $*$)

End of Pseudo - Code. [1]

2.3.2 Borderline-SMOTE

In order to achieve better prediction, most of the classification algorithms attempt to learn the borderline of each class as exactly as possible in the training process. The examples on the borderline and the ones nearby (we call them borderline examples in this paper) are more apt to be misclassified than the ones far from the borderline, and thus more important for classification.

Based on the analysis above, those examples far from the borderline may contribute little to classification. We thus present two new minority over-sampling methods, borderline-SMOTE1 and borderline-SMOTE2, in which only the borderline examples of the minority class are over-sampled. Our methods are different from the existing over-sampling methods in which all the minority examples or a random subset of the minority class are over-sampled [2, 5, 9] .

Our methods are based on SMOTE (Synthetic Minority Over-sampling Technique) [2]. SMOTE generates synthetic minority examples to over-sample the minority class. For every minority example, its k (which is set to 5 in SMOTE) nearest neighbors of the same class are calculated, then some examples are randomly selected from them according to the over-sampling rate. After that, new synthetic examples are generated along the line between the minority example and its selected nearest neighbors. Not like the existing over-sampling methods, our methods only oversample or strengthen the borderline minority examples. First, we find out the border-line minority examples; then, synthetic examples are generated from them and added to the original training set. Suppose that the whole training set is T, the minority class is P and the majority class is N, and

$$\text{P}=\{p_1, p_2, ..., p_{pnum}\}, N = \{n_1, n_2, ..., n_{nnum}\}$$

where *pnum* and *nnum* are the number of minority and majority examples. The detailed procedure of borderline-SMOTE1 is as follows.

Step.1 For every p_i $(i = 1, 2, ..., pnum)$ *in the minority class P,we calculate its m nearest neighbors from the whole training set T. The number of majority examples among the m nearest neighbors is denoted by m'(0 ≤m'≤m)*

Step 2. If m'=m i.e. all the m nearest neighbors of p_i *are majority examples,*p_i *is considered to be noise and is not operated in the following steps. If m/2≤m'<m,namely the number of* p_i'*smajority nearest neighbors is larger than the number of its minority ones,*p_i *is considered to be easily misclassified and put into a set DANGER. If 0≤m'<m,*p_i *is safe and needs not to participate in the follows steps.*

Step 3. The examples in DANGER are the borderline data of the minority class P, and we can see that P DANGER ⊆P.we set

$$DANGER = \{p_1', p_2', ..., p_{dnum}'\},\ \ 0 \leq dnum \leq pnum$$

For each example in *DANGER*, we calculate its *k*nearest neighbors from *P*

Step 4. In this step, we generate s×dnum synthetic positive examples from the data in DANGER, where s is an integer between 1 and k . For each p_i'*, we randomly select s nearest neighbors from its k nearest neighbors in P. Firstly, we calculate the differences* dif_j*(j=1,2,...,s) between* p_i' *and its s nearest neighbors from P , then multiply*dif_j *by a random number* r_j*(j =1,2,...,s) between 0 and 1, finally, s new synthetic minority examples are generated between* p_i' *and its nearest neighbors:*

$$synthetic_j = p'_i + r_j \times dif_j, \quad j=1,2,...,s$$

We repeat the above procedure for each p'_i in DANGER and can attain $s \times dnum$synthetic examples. This step is similar with SMOTE, for more detail see.

In the procedure above p_i, n_i, p'_i, dif_j and $synthetic_j$ are vectors. We can see that new synthetic data are generated along the line between the minority borderline examples and their nearest neighbors of the same class, thus strengthened the borderline examples.

Borderline-SMOTE2 not only generates synthetic examples from each example in *DANGER* and its positive nearest neighbors in *P*,but also does that from its nearest negative neighbor in *N*. The difference between it and its nearest negative neighbor is multiplied a random number between 0 and 0.5, thus the new generated examples are closer to the minority class [7].

2.3.3 Safe-level-SMOTE

Based on SMOTE, Safe-Level-SMOTE, Safe-Level-Synthetic Minority Oversampling TEchnique, assigns each positive instance its safe level before generating synthetic instances. Each synthetic instance is positioned closer to the largest safe level so all synthetic instances are generated only in safe regions. The safe level (sl) is defined as formula (1). If the safe level of an instance is close to 0, the instance is nearly noise. If it is close to k, the instance is considered safe. The safe level ratio is defined as formula (2). It is used for selecting the safe positions to generate synthetic instances.

safe level (sl)=the number of a positive stances in k nearest neighbours. (1)

safe level ratio = sl of a positive instance / sl of a nearest neighbours. (2)

Safe-Level-SMOTE algorithm

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. p, n and s are vectors. sl_p , sl_n , sl_ratio , *numattrs* , *dif* , and *gap* are scalars.

After assigning the *safe level* l to p and the *safe level* to n, the algorithm calculates the *safe level ratio*. There are five cases corresponding to the value of *safe level ratio* showed in the lines 12 to 28 of algorithm.

The first case showed in the lines 12 to 14 of algorithm. 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 noises. 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 showed in the lines 17 to 19 of algorithm. The *safe level ratio* is equal to ∞ and the *safe level* of p is not equal to 0. It means that n is noise. 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 showed in the lines 20 to 22 of algorithm. 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.

The fourth case showed in the lines 23 to 25 of algorithm. 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 because p is safer than n. The synthetic instance will be generated in the range [0, 1 / *safe level ratio*].

The fifth case showed in the lines 26 to 28 of algorithm. 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 because n is safer than p. The synthetic instance will be generated in the range [1 - *safe level ratio*, 1].

After each iteration of *for* loop in line 2 finishes, if the first case does not occurs, a synthetic instance s will be generated along the specific-ranged line 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 instances that satisfy the first case.

Algorithm: Safe-Level-SMOTE

Input: a set of all original positive instances D

Output: a set of all synthetic positive instances D'

1. D'= $\emptyset$
2. for each positive instance p in D
3. compute k nearest neighbours for p in D and

randomly select one from the k nearest neighbours, call it n

4. sl_p= the number of positive stances in k nearest neighbours for p in D
5. sl_n= the number of positive stances in k nearest neighbours for n in D
6. if ($sl_n \neq 0$); *sl is safe level.*
7. $sl_ratio = sl_p/sl_n$; sl_ratio *is safe level ratio.*
8. }
9. else{
10. $sl_ratio = \infty$
11. }
12. if ($sl_ratio = \infty$ AND $sl_p = 0$) { ; *the* 1^{st} *case*
13. does not generate positive synthetic instance
14. }
15. else{
16. for (atti = 1 to numattrs) { ; *numattrs is the number of attributes.*
17. if ($sl_ratio = \infty$ AND $sl_p \neq 0$) { ; *the* 2^{nd} *case*
18. gap=0
19. }

20. else if ($sl_ratio = 1$){ ; *the* $3th$ *case*

21. random a number between 0 and 1, call it gap

22. }

23. else if ($sl_ratio > 1$){ ; *the* $4th$ *case*

24. random a number between 0 and $1/sl_ratio$, call it gap

25. }

26. else if ($sl_ratio < 1$){ ; *the* $5th$ *case*

27. random a number between 1-sl_ratio, call it gap

28. }

29. dif = n[atti] - p[atti]

30. s[atti] = p[atti]+gap·dif

31. }

32. D' = D'∪ s

33. }

34. }

35. return D'

[4]

Chapter 3

Study cases and Discussion

In this study, we investigate the concept, extract the technique and evaluate the performance of the family of SMOTE algorithm, such as, SMOTE,Borderline-SMOTE and Safe-Level SMOTE

3.1 SMOTE (Synthetic Minority Over-sampling Technique)

The SMOTE is a technique to increase the amount of synthetics data that is in the minority of classes. The data was randomly selected from the original data or reconstructed from existing examples. Therefore, we believe that the classification model is of interest to the minority class. And the effect of class division will makes the data more balanced.

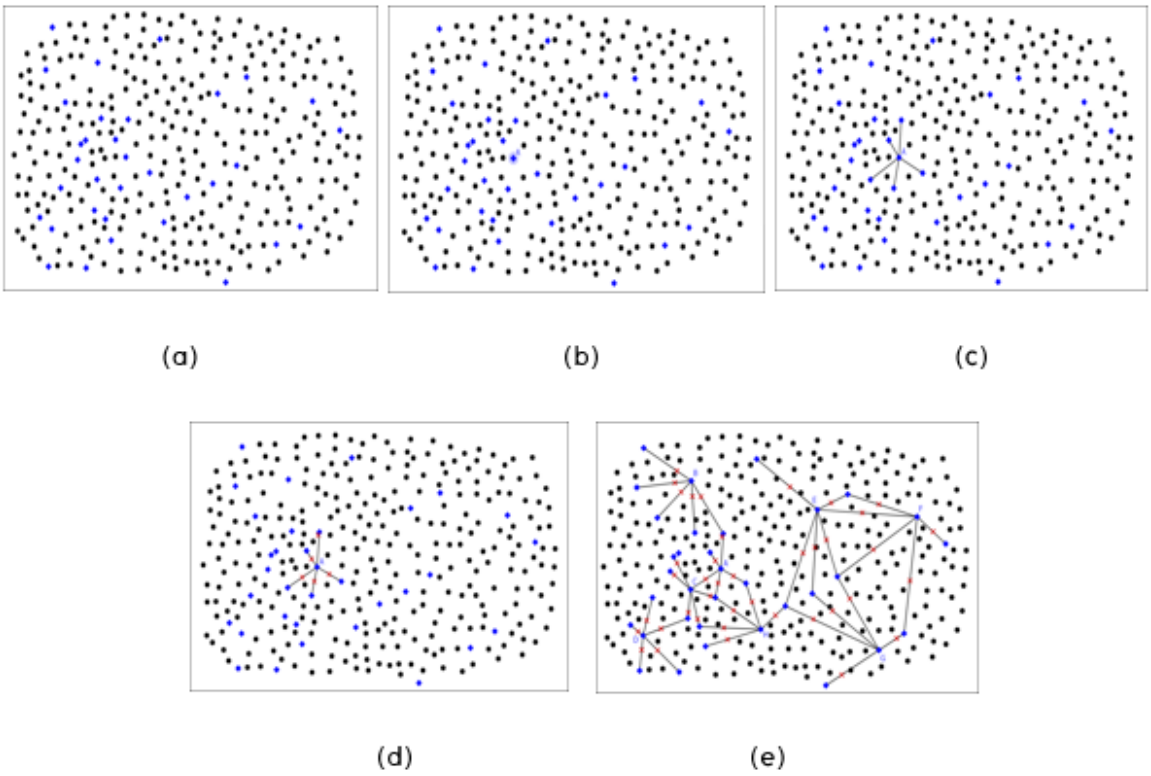


Figure 3.1: SMOTE Operation Procedure

Fig.3.1(a) Displays the distribution of data. Fig.3.1(b) Selected point of the minority class and called A.In Fig.3.1(c) selected k nearest neighbor in the minority to generate a synthetic instance, where k = 5. Fig.3.1(d) illustrates the generates synthetic instance, where a red cross represents a synthetic instance. Fig.3.1(e) The synthetics instance of SMOTE Technique.

From algorithm Choose a random number between 1 and k, call it nn.

Choose one of the k nearest neighbors of i

For 1 attribute

Compute: dif = $Sample[nn][attr]$ - $Sample[i][attr]$

Compute: gap = random number between 0 and 1

$Synthetic[attr] = Sample[i][attr] + gap * dif$

Example 3.1. $Sample[i][attr]$; A, A = (4.42,5.47)

$Sample[nn][attr]$; B, B = (4.52,6.66)

There fore ; dif = 4.52 - 4.42 = 0.1 and dif = 6.66 - 5.47 = 1.19

$gap = 0.5$

So ; $Synthetic[attr] = 4.42 + (0.5)(0.1) = 4.47$ and

$Synthetic[attr] = 5.47 + (0.5)(1.19) = 6.57$

Thus ; $Synthetic[attr] = (4.47,6.57)$

The synthetic instances is generated until the number of synthetic instances is equal to the setting a new point and continue to synthetic until the end. As shown in Fig. 3.1(e)

Advantages

SMOTE is a technique to increase the amount of data in a minority class to increase the number and make the data more balanced.

Disadvantages

SMOTE generates the same number of synthetic data samples for each original minority example and does so without consideration to neighboring examples, which increases the occurrence of over-lapping between classes.

For this reason, an adaptive sampling method has been proposed to correct this limitation. Borderline-SMOTE Show that the method of Borderline-SMOTE Give better results than SMOTE.

3.2 Borderline-SMOTE

Borderline-SMOTE divided positive instances into three points; noise, borderline, and safe in Fig. 3.2 Borderline-SMOTE generate synthetic instance only borderline point. borderline-SMOTE uses the same over-sampling technique as SMOTE but it over-samples only the borderline instances of a minority class.

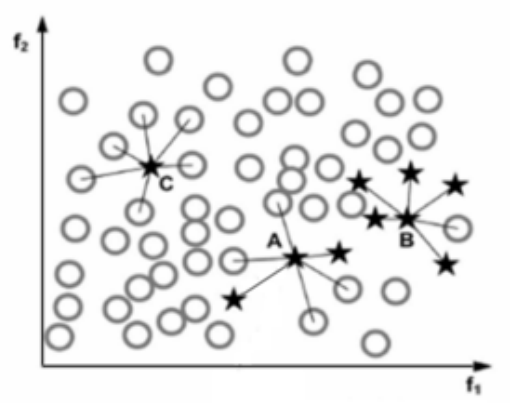


Figure 3.2: A is borderline point. B is safe point. C is noise point.

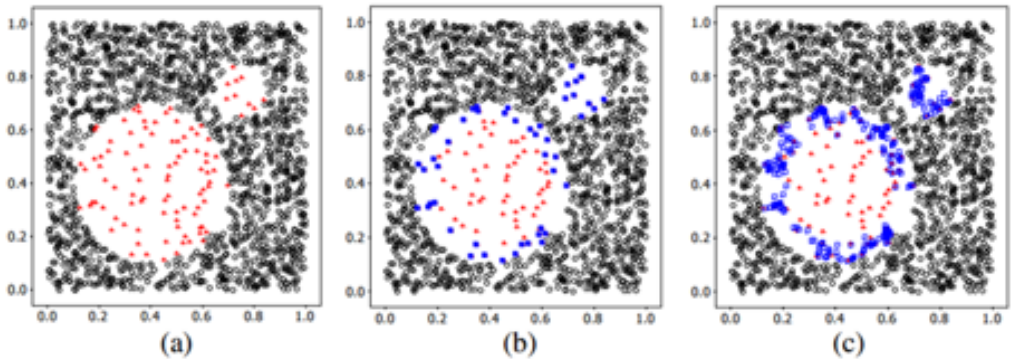


Figure 3.3: Borderline-SMOTE Operation Procedure

Fig.3.3(a) The original distribution of Circle data set. Fig.3.3(b). The border mi-

nority examples (solid squares). Fig.3.3(c) The borderline synthetic minority examples (hollow squares).

From algorithm s ; integer between 1 and k, where k is k- nearest neighbors

p'; randomly select from k-nearest neighbors

dif ; between p' and s nearest neighbors

r ; select between 0 - 1

$synthetic = p' + r$ x *dif*

Example 3.2 Point M (p')= (1.5,9.6)

Point N = (1.19,8.23)

dif = 1.19 - 1.5 = -0.3 and 8.23 - 9.6 = -1.37

$r = 0.5$

synthetic = 1.5 + (0.5)(-0.3) = 1.35 and 9.6 + (0.5)(-1.37) = 8.92

Thus ; *synthetic* = (1.35,8.92)

Advantages

Borderline-SMOTE generates synthetic case only for those minority samples that are "closer" to the border only, while SMOTE generates synthetic instances of all areas, regardless of the overlap of the class.

Disadvantages

A classifier is more likely to mis-detect. Because synthetic is close to majority class, while Safe-Level-SMOTE operates throughout a dataset and positions synthetic instances close to a safe region. Therefore, the case will to less overlap.

3.3 Safe-Level-SMOTE

Safe-Level-SMOTE, Safe-Level-Synthetic Minority Oversampling TEchnique, assigns each positive instance its safe level before generating synthetic instances. Each synthetic instance is positioned closer to the largest safe level so all synthetic instances are generated only in safe regions.

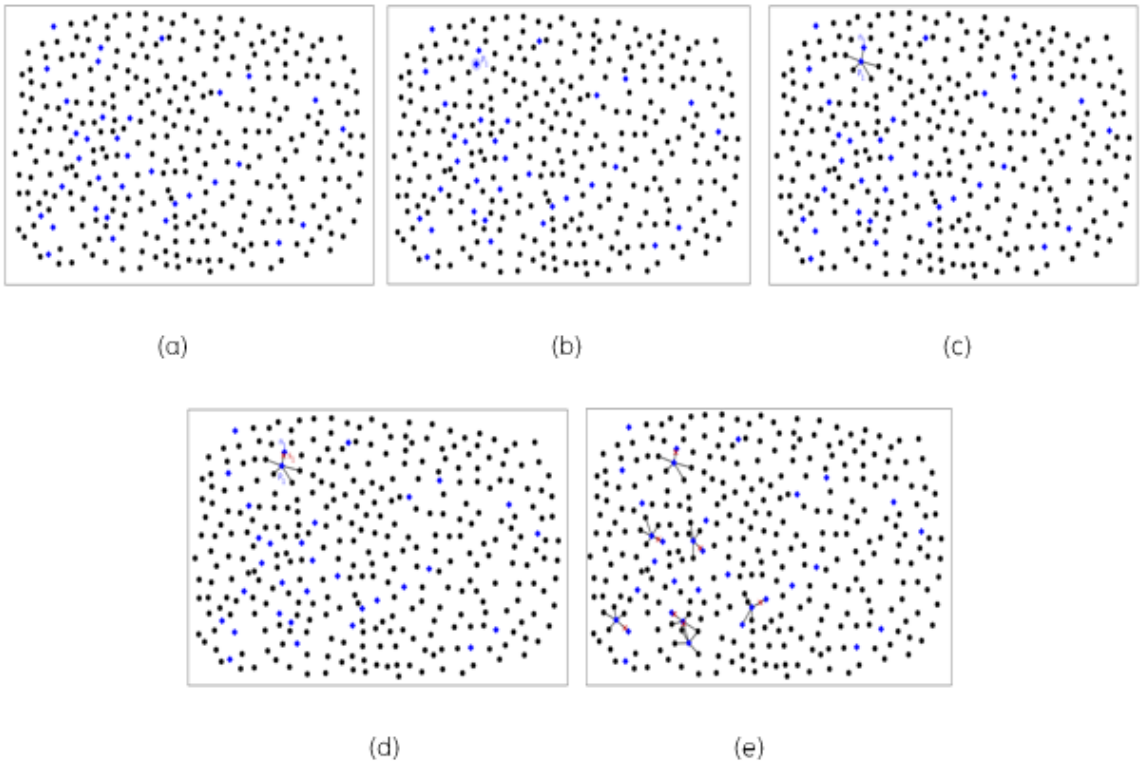


Figure 3.4: Safe-Level-SMOTE Operation Procedure

Fig.3.4(a) Displays the distribution of data. Fig.3.4(b) Selected point of the minority class and called P1. Fig.3.4(c) selected k nearest neighbor to generate a synthetic instance, where k = 5, And select a point in minority class is nearest to P1 and is called n1. Fig.3.4(d) illustrates the generates synthetic instance, where a red cross represents

a synthetic instance. Fig.3.4(e) The synthetics instance of Safe-Level-SMOTE.

From algorithm for each positive instance p

compute k nearest neighbours for p

randomly select one from the k nearest neighbours, call it n

sl_p = the number of positive stances in k nearest neighbours for p

sl_n = the number of positive stances in k nearest neighbours for n

sl_ratio = sl_p/sl_n ; sl_ratio is safe level ratio.

For 1 attribute

if sl_ratio = 1 random a number between 0 and 1, call it *gap*

if sl_ratio > 1 random a number between 0 and 1/sl_ratio, call it *gap*

if sl_ratio < 1 random a number between 1-sl_ratio and 1, call it *gap*

$dif = n[atti]$ - $p[atti]$

$s[atti] = p[atti] + gap * dif$, s is synthetic positive instances

Example 3.3 $p[atti]$ = (1.98,7.2)

$n[atti]$ = (1.19,8.23)

$sl_p = 1$

$sl_n = 4$

sl_ratio = 1/4 = 0.25 < 1

if sl_ratio < 1 random a number between 1-sl_ratio and 1, call it *gap*

gap = 1 - 0.25 = 0.75 random a number between 0.75 and 1

select *gap* = 0.8

dif = 1.19 - 1.98 = -0.79 and 8.23 - 7.2 = 1.03

$$s[atti] = 1.98 + (0.8)(-0.79) = 1.35 \text{ and } 7.2 + (0.8)(1.03) = 8.02$$

Thus ; synthetic = (1.35,8.02)

The synthetic instances is generated until the number of synthetic instances is equal to the setting a new point and continue to synthetic until the end. As shown in Fig. 3.4(e).

Advantages

Safe-Level-SMOTE carefully over-samples a dataset. Each synthetic instance is generated in safe position by considering the safe level ratio of instances. In contrast, SMOTE and Borderline-SMOTE may generate synthetic instances in unsuitable locations, such as overlapping regions and noise regions.

Disadvantages

Safe-Level-SMOTE does not concentrate on dense regions with positive instances. So, the classifier disregard this area.

Chapter 4

Conclusion

Attributes	Method		
	SMOTE	Borderline–SMOTE	Safe–Level–SMOTE
1. Configure k	✓	✓	✓
2. k nearest neighbor take only minority class	✓		
3. Divide the safe region into 3 levels		✓	✓
4. Calculate the 3 regions			
5. Calculates 2 levels of space			✓
6. Calculates 1 levels of space		✓	
7. Synthetic along the lines created	✓		
8. Compute $S = x_1 + gap \cdot dif$ $S = Y_1 + gap \cdot dif$	✓	✓	✓
9. gap = random number between [0,1]	✓	✓	✓
10. Over–lapping	✓	✓	✓
11. Speed of creation algorithm (BigO)	$O(n^2)$	$O(n^2)$	$O(n^2)$

Figure 4.1: Comparative tables of SMOTE family such as SMOTE, Borderline-SMOTE and Safe-Level-SMOTE.

Figure 4.1 shows a comparison of the performance of SMOTE techniques, Borderline-SMOTE techniques, Safe-Level-SMOTE techniques. We see that SMOTE techniques, Borderline-SMOTE techniques, and Safe-Level-SMOTE techniques are defined the nearest neighbor (k). and SMOTE techniques catch the minority class only. For Borderline-SMOTE techniques and Safe-Level-SMOTE techniques are divide the safe region into 3 levels and found no algorithms that computes 3 regions because we cut out the regions called noise which Safe-Level-SMOTE techniques calculates 2 levels of space that is very safe and less safe and Borderline- SMOTE techniques calculates 1 levels of space that is borderline region. The SMOTE techniques would be synthetic for all k but Borderline-SMOTE and Safe-Level-SMOTE are only synthetic for some lines. The three methods have the same formula for calculating s, choose a gap with a random number between [0,1], occur over-lapping and have the same speed that called "$O(n^2)$".

Data mining is a process that involves a lot of data. At present, data mining has been applied in many types of work whether it be business, management, science and medicine. How can data mining be applied to other areas?. We consider the medical field, it is found the information of patients with various diseases. A lot is also imbalanced information as well. Because of the number of patients, there may be a greater number of people with the flu than the number of patients with allergies. At present, we find a lot of imbalanced data. The number of data in one class is less than the data in another class. There are several techniques that will help solve this imbalanced data. One technique of data mining called Classification. Classification techniques is a classification of information to use as a model to predict the information we are interested. When we take this classified information into consideration. There

are problems in classification. Because classifiers are more interested in the majority class than the minority class, these minority class may be important that we cannot ignore them and to solve this problem. We use the popular technique called SMOTE. SMOTE uses over-sampling techniques to increase the number of minority class. But also problems with this technique, which are overlapping problems. Later SMOTE was developed as a technique. Borderline-SMOTE which is the technique used. Over-sampling same with SMOTE and even Borderline-SMOTE try to solve the over-lapping problem. But cannot solve this problem completely and when this technique was studied, it was found. There are overlapping problems, but they are overlapping with SMOTE techniques. Borderline-SMOTE that is Safe-level-SMOTE a technique was developed to solve the same overlap problem. This technique has a very high level of security in producing synthetic samples. We study this technique, it is safe-level-SMOTE can solve the over-lapping problem. But it is not perfect,but the over-lapping problem is considerably reduced compared to SMOTE and Borderline-SMOTE.

References

[1] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[2] Kegelmeyer W.P. Chawla, N.V., Bowyer,K.W., Hall, L.O. SMOTE: Synthetic Minority Over-Sampling Technique. Journal of Artificial Intelligence Research. pages 321–357, 2002.

[3] Kittipong Chomboon. Classification technique for minority class on imbalanced dataset with data partitioning method. *Suranaree University of Technology*, 2015.

[4] Chumphol Bunkhumpornpat, Krung Sinapiromsaran and Chidchanok Lursinsap. Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem. *Springer-Verlag Berlin Heidelberg 2009*, pages pp 475–482, 2009.

[5] G. Weiss. Mining with rarity: A unifying framework. *SIGKDD Explorations*, 6(1): 7–19, 2004.

[6] Micheline Kamber Han, Jiawei. *Data mining concepts and techniques*. Morgan Kaufman Publishers, United States of America, second edition, 2006.

[7] Hui Han , Wen-Yuan Wang and Bing-Huan Mao. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. *Springer-Verlag Berlin Heidelberg 2005*, pages pp 878 – 887, 2005.

[8] MlR. Imbalanced Classification Problems.

[9] Nathalie Japkowicz Nitesh V.Chawla and Aleksander Kolcz. Editorial: Special Issue on Learning from Imbalanced Data Sets. *SIGKDD Explorations*, 6(1):1–6, 2004.

[10] Paranya Palwisut. Improving Decision Tree Technique in Imbalanced Data Sets Using SMOTE for Internet Addiction Disorder Data. *Information Technology Journal*, pages 54–63, 2016.

[11] Upasana. How to handle Imbalanced Classification Problems in machine learning?, 2017.

[12] Wikipedia. Oversampling and undersampling in data analysis, 2011.

.

Appendix

.

Appendix A : SMOTE

Journal of Artificial Intelligence Research 16 (2002) 321–357 Submitted 09/01; published 06/02

SMOTE: Synthetic Minority Over-sampling Technique

Nitesh V. Chawla CHAWLA@CSEE.USF.EDU
Department of Computer Science and Engineering, ENB 118
University of South Florida
4202 E. Fowler Ave.
Tampa, FL 33620-5399, USA

Kevin W. Bowyer KWB@CSE.ND.EDU
Department of Computer Science and Engineering
384 Fitzpatrick Hall
University of Notre Dame
Notre Dame, IN 46556, USA

Lawrence O. Hall HALL@CSEE.USF.EDU
Department of Computer Science and Engineering, ENB 118
University of South Florida
4202 E. Fowler Ave.
Tampa, FL 33620-5399, USA

W. Philip Kegelmeyer WPK@CALIFORNIA.SANDIA.GOV
Sandia National Laboratories
Biosystems Research Department, P.O. Box 969, MS 9951
Livermore, CA, 94551-0969, USA

Abstract

An approach to the construction of classifiers from imbalanced datasets is described. A dataset is imbalanced if the classification categories are not approximately equally represented. Often real-world data sets are predominately composed of "normal" examples with only a small percentage of "abnormal" or "interesting" examples. It is also the case that the cost of misclassifying an abnormal (interesting) example as a normal example is often much higher than the cost of the reverse error. Under-sampling of the majority (normal) class has been proposed as a good means of increasing the sensitivity of a classifier to the minority class. This paper shows that a combination of our method of over-sampling the minority (abnormal) class and under-sampling the majority (normal) class can achieve better classifier performance (in ROC space) than only under-sampling the majority class. This paper also shows that a combination of our method of over-sampling the minority class and under-sampling the majority class can achieve better classifier performance (in ROC space) than varying the loss ratios in Ripper or class priors in Naive Bayes. Our method of over-sampling the minority class involves creating synthetic minority class examples. Experiments are performed using C4.5, Ripper and a Naive Bayes classifier. The method is evaluated using the area under the Receiver Operating Characteristic curve (AUC) and the ROC convex hull strategy.

1. Introduction

A dataset is imbalanced if the classes are not approximately equally represented. Imbalance on the order of 100 to 1 is prevalent in fraud detection and imbalance of up to 100,000 to

1 has been reported in other applications (Provost & Fawcett, 2001). There have been attempts to deal with imbalanced datasets in domains such as fraudulent telephone calls (Fawcett & Provost, 1996), telecommunications management (Ezawa, Singh, & Norton, 1996), text classification (Lewis & Catlett, 1994; Dumais, Platt, Heckerman, & Sahami, 1998; Mladenić & Grobelnik, 1999; Lewis & Ringuette, 1994; Cohen, 1995a) and detection of oil spills in satellite images (Kubat, Holte, & Matwin, 1998).

The performance of machine learning algorithms is typically evaluated using predictive accuracy. However, this is not appropriate when the data is imbalanced and/or the costs of different errors vary markedly. As an example, consider the classification of pixels in mammogram images as possibly cancerous (Woods, Doss, Bowyer, Solka, Priebe, & Kegelmeyer, 1993). A typical mammography dataset might contain 98% normal pixels and 2% abnormal pixels. A simple default strategy of guessing the majority class would give a predictive accuracy of 98%. However, the nature of the application requires a fairly high rate of correct detection in the minority class and allows for a small error rate in the majority class in order to achieve this. Simple predictive accuracy is clearly not appropriate in such situations. The Receiver Operating Characteristic (ROC) curve is a standard technique for summarizing classifier performance over a range of tradeoffs between true positive and false positive error rates (Swets, 1988). The Area Under the Curve (AUC) is an accepted traditional performance metric for a ROC curve (Duda, Hart, & Stork, 2001; Bradley, 1997; Lee, 2000). The ROC convex hull can also be used as a robust method of identifying potentially optimal classifiers (Provost & Fawcett, 2001). If a line passes through a point on the convex hull, then there is no other line with the same slope passing through another point with a larger true positive (TP) intercept. Thus, the classifier at that point is optimal under any distribution assumptions in tandem with that slope.

The machine learning community has addressed the issue of class imbalance in two ways. One is to assign distinct costs to training examples (Pazzani, Merz, Murphy, Ali, Hume, & Brunk, 1994; Domingos, 1999). The other is to re-sample the original dataset, either by over-sampling the minority class and/or under-sampling the majority class (Kubat & Matwin, 1997; Japkowicz, 2000; Lewis & Catlett, 1994; Ling & Li, 1998). Our approach (Chawla, Bowyer, Hall, & Kegelmeyer, 2000) blends under-sampling of the majority class with a special form of over-sampling the minority class. Experiments with various datasets and the C4.5 decision tree classifier (Quinlan, 1992), Ripper (Cohen, 1995b), and a Naive Bayes Classifier show that our approach improves over other previous re-sampling, modifying loss ratio, and class priors approaches, using either the AUC or ROC convex hull.

Section 2 gives an overview of performance measures. Section 3 reviews the most closely related work dealing with imbalanced datasets. Section 4 presents the details of our approach. Section 5 presents experimental results comparing our approach to other re-sampling approaches. Section 6 discusses the results and suggests directions for future work.

2. Performance Measures

The performance of machine learning algorithms is typically evaluated by a confusion matrix as illustrated in Figure 1 (for a 2 class problem). The columns are the Predicted class and the rows are the Actual class. In the confusion matrix, TN is the number of negative examples

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

Figure 1: Confusion Matrix

correctly classified (True Negatives), FP is the number of negative examples incorrectly classified as positive (False Positives), FN is the number of positive examples incorrectly classified as negative (False Negatives) and TP is the number of positive examples correctly classified (True Positives).

Predictive accuracy is the performance measure generally associated with machine learning algorithms and is defined as $Accuracy = (TP + TN)/(TP + FP + TN + FN)$. In the context of balanced datasets and equal error costs, it is reasonable to use error rate as a performance metric. Error rate is $1 - Accuracy$. In the presence of imbalanced datasets with unequal error costs, it is more appropriate to use the ROC curve or other similar techniques (Ling & Li, 1998; Drummond & Holte, 2000; Provost & Fawcett, 2001; Bradley, 1997; Turney, 1996).

ROC curves can be thought of as representing the family of best decision boundaries for relative costs of TP and FP. On an ROC curve the X-axis represents $\%FP = FP/(TN+FP)$ and the Y-axis represents $\%TP = TP/(TP+FN)$. The ideal point on the ROC curve would be (0,100), that is all positive examples are classified correctly and no negative examples are misclassified as positive. One way an ROC curve can be swept out is by manipulating the balance of training samples for each class in the training set. Figure 2 shows an illustration. The line y = x represents the scenario of randomly guessing the class. Area Under the ROC Curve (AUC) is a useful metric for classifier performance as it is independent of the decision criterion selected and prior probabilities. The AUC comparison can establish a dominance relationship between classifiers. If the ROC curves are intersecting, the total AUC is an average comparison between models (Lee, 2000). However, for some specific cost and class distributions, the classifier having maximum AUC may in fact be suboptimal. Hence, we also compute the ROC convex hulls, since the points lying on the ROC convex hull are potentially optimal (Provost, Fawcett, & Kohavi, 1998; Provost & Fawcett, 2001).

3. Previous Work: Imbalanced datasets

Kubat and Matwin (1997) selectively under-sampled the majority class while keeping the original population of the minority class. They have used the geometric mean as a performance measure for the classifier, which can be related to a single point on the ROC curve. The minority examples were divided into four categories: some noise overlapping the positive class decision region, borderline samples, redundant samples and safe samples. The borderline examples were detected using the Tomek links concept (Tomek, 1976). Another

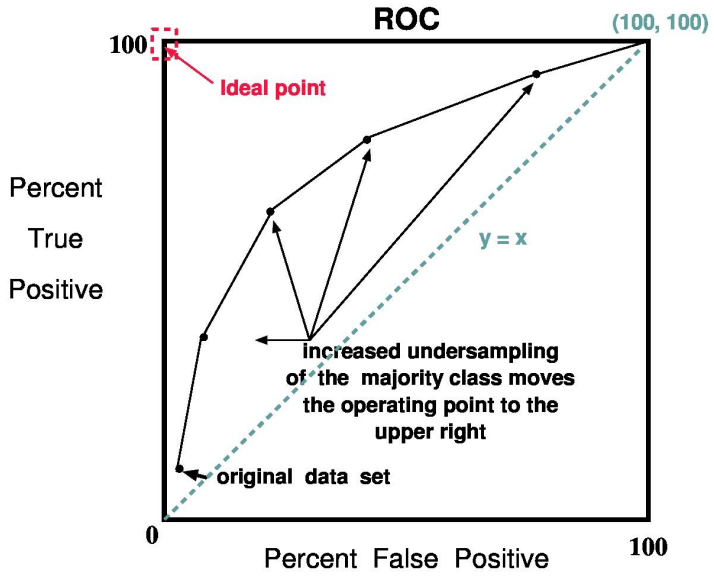


Figure 2: Illustration of sweeping out a ROC curve through under-sampling. Increased under-sampling of the majority (negative) class will move the performance from the lower left point to the upper right.

related work proposed the SHRINK system that classifies an overlapping region of minority (positive) and majority (negative) classes as positive; it searches for the "best positive region" (Kubat et al., 1998).

Japkowicz (2000) discussed the effect of imbalance in a dataset. She evaluated three strategies: under-sampling, resampling and a recognition-based induction scheme. We focus on her sampling approaches. She experimented on artificial 1D data in order to easily measure and construct concept complexity. Two resampling methods were considered. Random resampling consisted of resampling the smaller class at random until it consisted of as many samples as the majority class and "focused resampling" consisted of resampling only those minority examples that occurred on the boundary between the minority and majority classes. Random under-sampling was considered, which involved under-sampling the majority class samples at random until their numbers matched the number of minority class samples; focused under-sampling involved under-sampling the majority class samples lying further away. She noted that both the sampling approaches were effective, and she also observed that using the sophisticated sampling techniques did not give any clear advantage in the domain considered (Japkowicz, 2000).

One approach that is particularly relevant to our work is that of Ling and Li (1998). They combined over-sampling of the minority class with under-sampling of the majority class. They used lift analysis instead of accuracy to measure a classifier's performance. They proposed that the test examples be ranked by a confidence measure and then lift be used as the evaluation criteria. A lift curve is similar to an ROC curve, but is more tailored for the

marketing analysis problem (Ling & Li, 1998). In one experiment, they under-sampled the majority class and noted that the best lift index is obtained when the classes are equally represented (Ling & Li, 1998). In another experiment, they over-sampled the positive (minority) examples with replacement to match the number of negative (majority) examples to the number of positive examples. The over-sampling and under-sampling combination did not provide significant improvement in the lift index. However, our approach to over-sampling differs from theirs.

Solberg and Solberg (1996) considered the problem of imbalanced data sets in oil slick classification from SAR imagery. They used over-sampling and under-sampling techniques to improve the classification of oil slicks. Their training data had a distribution of 42 oil slicks and 2,471 look-alikes, giving a prior probability of 0.98 for look-alikes. This imbalance would lead the learner (without any appropriate loss functions or a methodology to modify priors) to classify almost all look-alikes correctly at the expense of misclassifying many of the oil slick samples (Solberg & Solberg, 1996). To overcome this imbalance problem, they over-sampled (with replacement) 100 samples from the oil slick, and they randomly sampled 100 samples from the non oil slick class to create a new dataset with equal probabilities. They learned a classifier tree on this balanced data set and achieved a 14% error rate on the oil slicks in a leave-one-out method for error estimation; on the look alikes they achieved an error rate of 4% (Solberg & Solberg, 1996).

Another approach that is similar to our work is that of Domingos (1999). He compares the "metacost" approach to each of majority under-sampling and minority over-sampling. He finds that metacost improves over either, and that under-sampling is preferable to minority over-sampling. Error-based classifiers are made cost-sensitive. The probability of each class for each example is estimated, and the examples are relabeled optimally with respect to the misclassification costs. The relabeling of the examples expands the decision space as it creates new samples from which the classifier may learn (Domingos, 1999).

A feed-forward neural network trained on an imbalanced dataset may not learn to discriminate enough between classes (DeRouin, Brown, Fausett, & Schneider, 1991). The authors proposed that the learning rate of the neural network be adapted to the statistics of class representation in the data. They calculated an attention factor from the proportion of samples presented to the neural network for training. The learning rate of the network elements was adjusted based on the attention factor. They experimented on an artificially generated training set and on a real-world training set, both with multiple (more than two) classes. They compared this to the approach of replicating the minority class samples to balance the data set used for training. The classification accuracy on the minority class was improved.

Lewis and Catlett (1994) examined heterogeneous uncertainty sampling for supervised learning. This method is useful for training samples with uncertain classes. The training samples are labeled incrementally in two phases and the uncertain instances are passed on to the next phase. They modified C4.5 to include a loss ratio for determining the class values at the leaves. The class values were determined by comparison with a probability threshold of $LR/(LR+1)$, where LR is the loss ratio (Lewis & Catlett, 1994).

The information retrieval (IR) domain (Dumais et al., 1998; Mladenić & Grobelnik, 1999; Lewis & Ringuette, 1994; Cohen, 1995a) also faces the problem of class imbalance in the dataset. A document or web page is converted into a bag-of-words representation;

that is, a feature vector reflecting occurrences of words in the page is constructed. Usually, there are very few instances of the interesting category in text categorization. This over-representation of the negative class in information retrieval problems can cause problems in evaluating classifiers' performances. Since error rate is not a good metric for skewed datasets, the classification performance of algorithms in information retrieval is usually measured by *precision* and *recall*:

$$recall = \frac{TP}{TP + FN}$$

$$precision = \frac{TP}{TP + FP}$$

Mladenić and Grobelnik (1999) proposed a feature subset selection approach to deal with imbalanced class distribution in the IR domain. They experimented with various feature selection methods, and found that the *odds ratio* (van Rijsbergen, Harper, & Porter, 1981) when combined with a Naive Bayes classifier performs best in their domain. *Odds ratio* is a probabilistic measure used to rank documents according to their relevance to the positive class (minority class). *Information gain* for a word, on the other hand, does not pay attention to a particular target class; it is computed per word for each class. In an imbalanced text dataset (assuming 98 to 99% is the negative class), most of the features will be associated with the negative class. *Odds ratio* incorporates the target class information in its metric giving better results when compared to *information gain* for text categorization.

Provost and Fawcett (1997) introduced the ROC convex hull method to estimate the classifier performance for imbalanced datasets. They note that the problems of unequal class distribution and unequal error costs are related and that little work has been done to address either problem (Provost & Fawcett, 2001). In the ROC convex hull method, the ROC space is used to separate classification performance from the class and cost distribution information.

To summarize the literature, under-sampling the majority class enables better classifiers to be built than over-sampling the minority class. A combination of the two as done in previous work does not lead to classifiers that outperform those built utilizing only under-sampling. However, the over-sampling of the minority class has been done by sampling with replacement from the original data. Our approach uses a different method of over-sampling.

4. SMOTE: Synthetic Minority Over-sampling TEchnique

4.1 Minority over-sampling with replacement

Previous research (Ling & Li, 1998; Japkowicz, 2000) has discussed over-sampling with replacement and has noted that it doesn't significantly improve minority class recognition. We interpret the underlying effect in terms of decision regions in feature space. Essentially, as the minority class is over-sampled by increasing amounts, the effect is to identify similar but more specific regions in the feature space as the decision region for the minority class. This effect for decision trees can be understood from the plots in Figure 3.

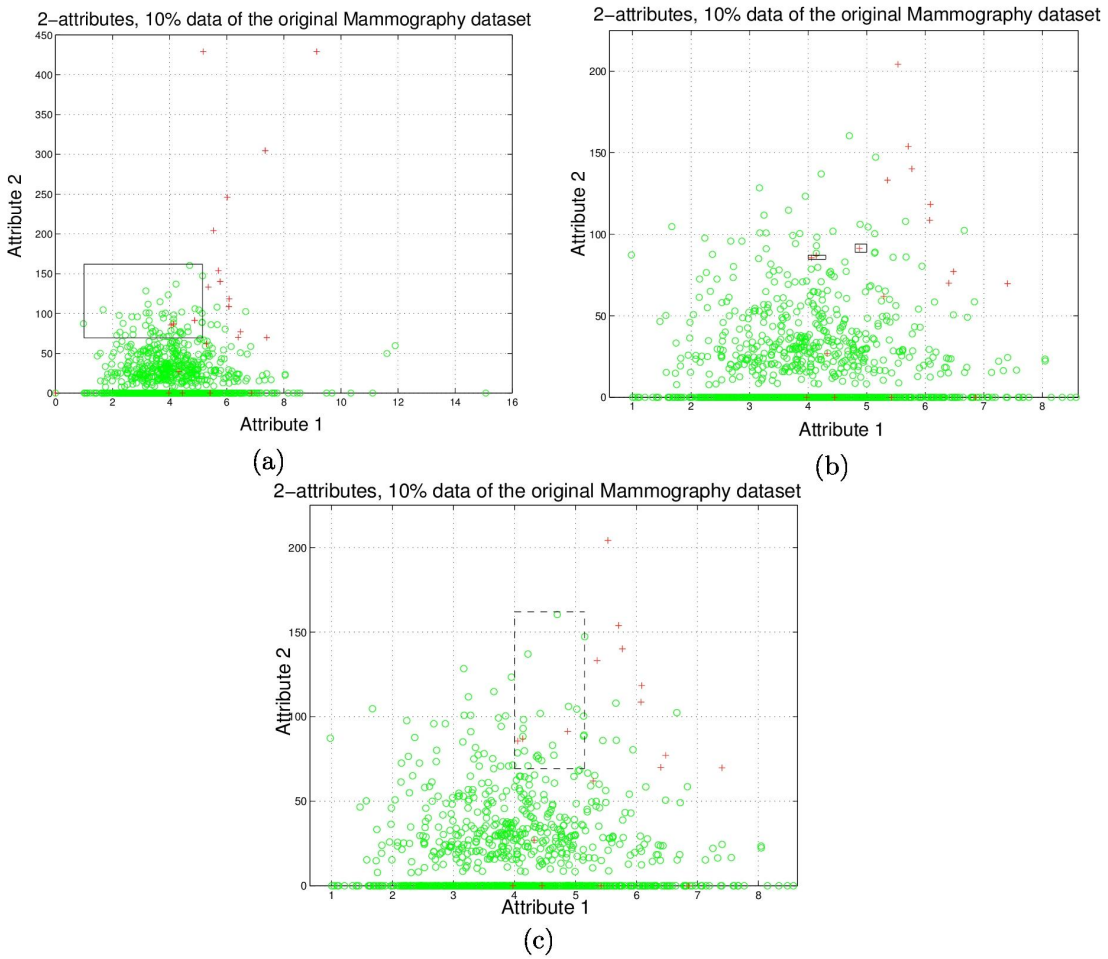


Figure 3: a) Decision region in which the three minority class samples (shown by '+') reside after building a decision tree. This decision region is indicated by the solid-line rectangle. b) A zoomed-in view of the chosen minority class samples for the same dataset. Small solid-line rectangles show the decision regions as a result of over-sampling the minority class with replication. c) A zoomed-in view of the chosen minority class samples for the same dataset. Dashed lines show the decision region after over-sampling the minority class with synthetic generation.

The data for the plot in Figure 3 was extracted from a Mammography dataset[1] (Woods et al., 1993). The minority class samples are shown by + and the majority class samples are shown by *o* in the plot. In Figure 3(a), the region indicated by the solid-line rectangle is a majority class decision region. Nevertheless, it contains three minority class samples shown by '+' as false negatives. If we replicate the minority class, the decision region for the minority class becomes very specific and will cause new splits in the decision tree. This will lead to more terminal nodes (leaves) as the learning algorithm tries to learn more and more specific regions of the minority class; in essence, overfitting. Replication of the minority class does not cause its decision boundary to spread into the majority class region. Thus, in Figure 3(b), the three samples previously in the majority class decision region now have very specific decision regions.

4.2 SMOTE

We propose an over-sampling approach in which the minority class is over-sampled by creating "synthetic" examples rather than by over-sampling with replacement. This approach is inspired by a technique that proved successful in handwritten character recognition (Ha & Bunke, 1997). They created extra training data by performing certain operations on real data. In their case, operations like rotation and skew were natural ways to perturb the training data. We generate synthetic examples in a less application-specific manner, by operating in "feature space" rather than "data space". The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen. Our implementation currently uses five nearest neighbors. For instance, if the amount of over-sampling needed is 200%, only two neighbors from the five nearest neighbors are chosen and one sample is generated in the direction of each. Synthetic samples are generated in the following way: Take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general.

Algorithm *SMOTE*, on the next page, is the pseudo-code for SMOTE. Table 4.2 shows an example of calculation of random synthetic samples. The amount of over-sampling is a parameter of the system, and a series of ROC curves can be generated for different populations and ROC analysis performed.

The synthetic examples cause the classifier to create larger and less specific decision regions as shown by the dashed lines in Figure 3(c), rather than smaller and more specific regions. More general regions are now learned for the minority class samples rather than those being subsumed by the majority class samples around them. The effect is that decision trees generalize better. Figures 4 and 5 compare the minority over-sampling with replacement and SMOTE. The experiments were conducted on the mammography dataset. There were 10923 examples in the majority class and 260 examples in the minority class originally. We have approximately 9831 examples in the majority class and 233 examples

1. The data is available from the USF Intelligent Systems Lab, http://morden.csee.usf.edu/~chawla.

in the minority class for the training set used in 10-fold cross-validation. The minority class was over-sampled at 100%, 200%, 300%, 400% and 500% of its original size. The graphs show that the tree sizes for minority over-sampling with replacement at higher degrees of replication are much greater than those for SMOTE, and the minority class recognition of the minority over-sampling with replacement technique at higher degrees of replication isn't as good as SMOTE.

Algorithm *SMOTE*(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE N%; Number of nearest neighbors k
Output: $(N/100)$ * T synthetic minority class samples
1. (* *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* *)
2. **if** $N < 100$
3. **then** Randomize the T minority class samples
4. $T = (N/100) * T$
5. $N = 100$
6. **endif**
7. $N = (int)(N/100)$ (* *The amount of SMOTE is assumed to be in integral multiples of 100.* *)
8. k = Number of nearest neighbors
9. $numattrs$ = Number of attributes
10. $Sample[\,][\,]$: array for original minority class samples
11. $newindex$: keeps a count of number of synthetic samples generated, initialized to 0
12. $Synthetic[\,][\,]$: array for synthetic samples
(* *Compute k nearest neighbors for each minority class sample only.* *)
13. **for** $i \leftarrow 1$ **to** T
14. Compute k nearest neighbors for i, and save the indices in the $nnarray$
15. Populate(N, i, $nnarray$)
16. **endfor**

Populate(N, i, $nnarray$) (* *Function to generate the synthetic samples.* *)
17. **while** $N \neq 0$
18. Choose a random number between 1 and k, call it nn. This step chooses one of the k nearest neighbors of i.
19. **for** $attr \leftarrow 1$ **to** $numattrs$
20. Compute: $dif = Sample[nnarray[nn]][attr] - Sample[i][attr]$
21. Compute: gap = random number between 0 and 1
22. $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
23. **endfor**
24. $newindex{+}{+}$
25. $N = N - 1$
26. **endwhile**
27. **return** (* *End of Populate.* *)
End of Pseudo-Code.

Consider a sample (6,4) and let (4,3) be its nearest neighbor.
(6,4) is the sample for which k-nearest neighbors are being identified.
(4,3) is one of its k-nearest neighbors.
Let:
f1_1 = 6 f2_1 = 4 f2_1 - f1_1 = -2
f1_2 = 4 f2_2 = 3 f2_2 - f1_2 = -1
The new samples will be generated as
(f1',f2') = (6,4) + rand(0-1) * (-2,-1)
rand(0-1) generates a random number between 0 and 1.

Table 1: Example of generation of synthetic examples (SMOTE).

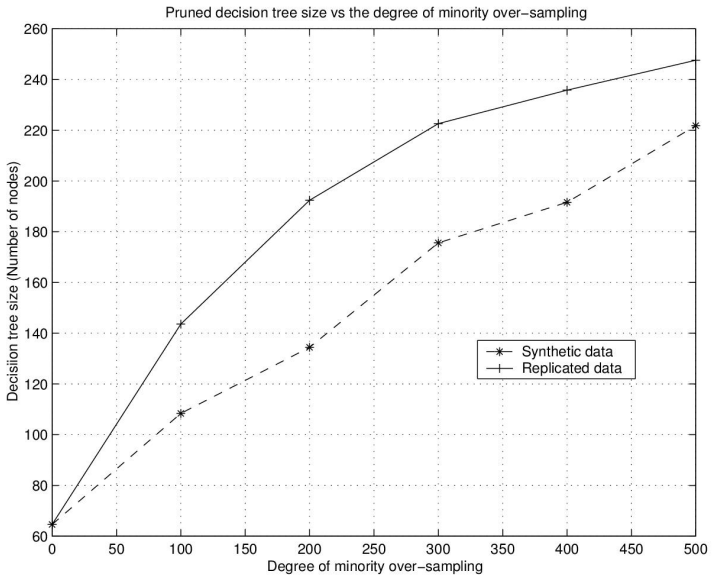


Figure 4: Comparison of decision tree sizes for replicated over-sampling and SMOTE for the Mammography dataset

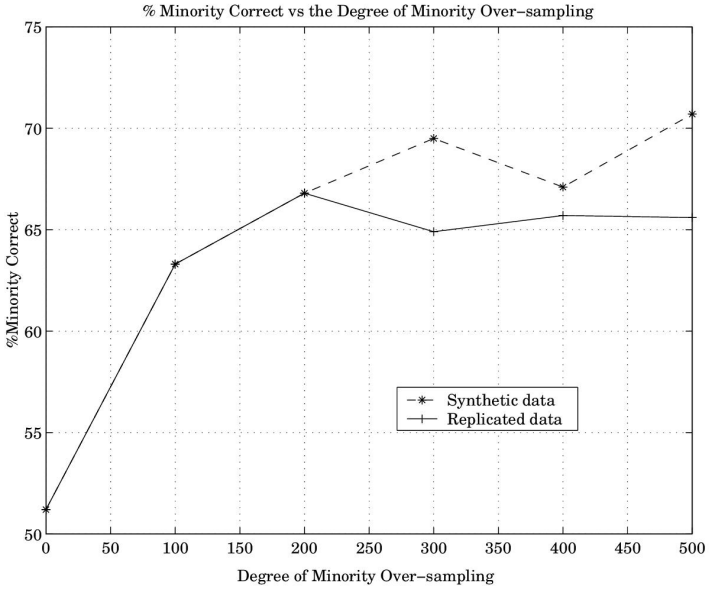


Figure 5: Comparison of % Minority correct for replicated over-sampling and SMOTE for the Mammography dataset

4.3 Under-sampling and SMOTE Combination

The majority class is under-sampled by randomly removing samples from the majority class population until the minority class becomes some specified percentage of the majority class. This forces the learner to experience varying degrees of under-sampling and at higher degrees of under-sampling the minority class has a larger presence in the training set. In describing our experiments, our terminology will be such that if we *under-sample the majority class at 200%*, it would mean that the modified dataset will contain *twice as many elements from the minority class as from the majority class*; that is, if the minority class had 50 samples and the majority class had 200 samples and we under-sample majority at 200%, the majority class would end up having 25 samples. By applying a combination of under-sampling and over-sampling, the initial bias of the learner towards the negative (majority) class is reversed in the favor of the positive (minority) class. Classifiers are learned on the dataset perturbed by "SMOTING" the minority class and under-sampling the majority class.

5. Experiments

We used three different machine learning algorithms for our experiments. Figure 6 provides an overview of our experiments.

1. **C4.5:** We compared various combinations of SMOTE and under-sampling with plain under-sampling using C4.5 release 8 (Quinlan, 1992) as the base classifier.

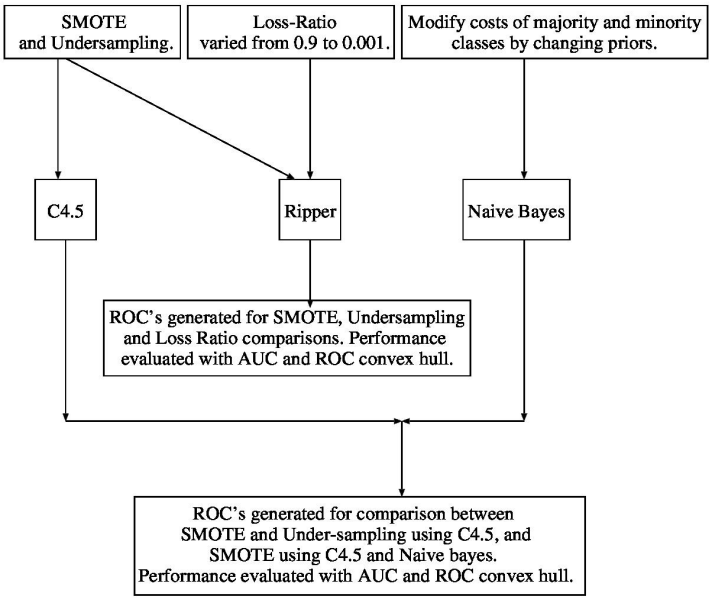


Figure 6: Experiments Overview

2. **Ripper:** We compared various combinations of SMOTE and under-sampling with plain under-sampling using Ripper (Cohen, 1995b) as the base classifier. We also varied Ripper's loss ratio (Cohen & Singer, 1996; Lewis & Catlett, 1994) from 0.9 to 0.001 (as a means of varying misclassification cost) and compared the effect of this variation with the combination of SMOTE and under-sampling. By reducing the loss ratio from 0.9 to 0.001 we were able to build a set of rules for the minority class.

3. **Naive Bayes Classifier:** The Naive Bayes Classifier[2] can be made cost-sensitive by varying the priors of the minority class. We varied the priors of the minority class from 1 to 50 times the majority class and compared with C4.5's SMOTE and under-sampling combination.

These different learning algorithms allowed SMOTE to be compared to some methods that can handle misclassification costs directly. %FP and %TP were averaged over 10-fold cross-validation runs for each of the data combinations. The minority class examples were over-sampled by calculating the five nearest neighbors and generating synthetic examples. The AUC was calculated using the trapezoidal rule. We extrapolated an extra point of TP = 100% and FP = 100% for each ROC curve. We also computed the ROC convex hull to identify the optimal classifiers, as the points lying on the hull are potentially optimal classifiers (Provost & Fawcett, 2001).

2. The source code was downloaded from http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html.

5.1 Datasets

We experimented on nine different datasets. These datasets are summarized in Table 5.2. These datasets vary extensively in their size and class proportions, thus offering different domains for SMOTE. In order of increasing imbalance they are:

1. The Pima Indian Diabetes (Blake & Merz, 1998) has 2 classes and 768 samples. The data is used to identify the positive diabetes cases in a population near Phoenix, Arizona. The number of positive class samples is only 268. Good sensitivity to detection of diabetes cases will be a desirable attribute of the classifier.

2. The Phoneme dataset is from the ELENA project[3]. The aim of the dataset is to distinguish between nasal (class 0) and oral sounds (class 1). There are 5 features. The class distribution is 3,818 samples in class 0 and 1,586 samples in class 1.

3. The Adult dataset (Blake & Merz, 1998) has 48,842 samples with 11,687 samples belonging to the minority class. This dataset has 6 continuous features and 8 nominal features. SMOTE and SMOTE-NC (see Section 6.1) algorithms were evaluated on this dataset. For SMOTE, we extracted the continuous features and generated a new dataset with only continuous features.

4. The E-state data[4] (Hall, Mohney, & Kier, 1991) consists of electrotopological state descriptors for a series of compounds from the National Cancer Institute's Yeast Anti-Cancer drug screen. E-state descriptors from the NCI Yeast AntiCancer Drug Screen were generated by Tripos, Inc. Briefly, a series of about 60,000 compounds were tested against a series of 6 yeast strains at a given concentration. The test was a high-throughput screen at only one concentration so the results are subject to contamination, etc. The growth inhibition of the yeast strain when exposed to the given compound (with respect to growth of the yeast in a neutral solvent) was measured. The activity classes are either active — at least one single yeast strain was inhibited more than 70%, or inactive — no yeast strain was inhibited more than 70%. The dataset has 53,220 samples with 6,351 samples of active compounds.

5. The Satimage dataset (Blake & Merz, 1998) has 6 classes originally. We chose the smallest class as the minority class and collapsed the rest of the classes into one as was done in (Provost et al., 1998). This gave us a skewed 2-class dataset, with 5809 majority class samples and 626 minority class samples.

6. The Forest Cover dataset is from the UCI repository (Blake & Merz, 1998). This dataset has 7 classes and 581,012 samples. This dataset is for the prediction of forest cover type based on cartographic variables. Since our system currently works for binary classes we extracted data for two classes from this dataset and ignored the rest. Most other approaches only work for only two classes (Ling & Li, 1998; Japkowicz, 2000; Kubat & Matwin, 1997; Provost & Fawcett, 2001). The two classes we considered are Ponderosa Pine with 35,754 samples and Cottonwood/Willow with 2,747

3. ftp.dice.ucl.ac.be in the directory pub/neural-nets/ELENA/databases.
4. We would like to thank Steven Eschrich for providing the dataset and description to us.

Dataset	Majority Class	Minority Class
Pima	500	268
Phoneme	3818	1586
Adult	37155	11687
E-state	46869	6351
Satimage	5809	626
Forest Cover	35754	2747
Oil	896	41
Mammography	10923	260
Can	435512	8360

Table 2: Dataset distribution

samples. Nevertheless, the SMOTE technique can be applied to a multiple class problem as well by specifying what class to SMOTE for. However, in this paper, we have focused on 2-classes problems, to explicitly represent positive and negative classes.

7. The Oil dataset was provided by Robert Holte and is used in their paper (Kubat et al., 1998). This dataset has 41 oil slick samples and 896 non-oil slick samples.

8. The Mammography dataset (Woods et al., 1993) has 11,183 samples with 260 calcifications. If we look at predictive accuracy as a measure of goodness of the classifier for this case, the default accuracy would be 97.68% when every sample is labeled non-calcification. But, it is desirable for the classifier to predict most of the calcifications correctly.

9. The Can dataset was generated from the Can ExodusII data using the AVATAR (Chawla & Hall, 1999) version of the Mustafa Visualization tool[5]. The portion of the can being crushed was marked as "very interesting" and the rest of the can was marked as "unknown." A dataset of size 443,872 samples with 8,360 samples marked as "very interesting" was generated.

5.2 ROC Creation

A ROC curve for SMOTE is produced by using C4.5 or Ripper to create a classifier for each one of a series of modified training datasets. A given ROC curve is produced by first over-sampling the minority class to a specified degree and then under-sampling the majority class at increasing degrees to generate the successive points on the curve. The amount of under-sampling is identical to plain under-sampling. So, each corresponding point on each ROC curve for a dataset represents the same number of majority class samples. Different ROC curves are produced by starting with different levels of minority over-sampling. ROC curves were also generated by varying the loss ratio in Ripper from 0.9 to 0.001 and by varying the priors of the minority class from the original distribution to up to 50 times the majority class for a Naive Bayes Classifier.

5. The Mustafa visualization tool was developed by Mike Glass of Sandia National Labs.

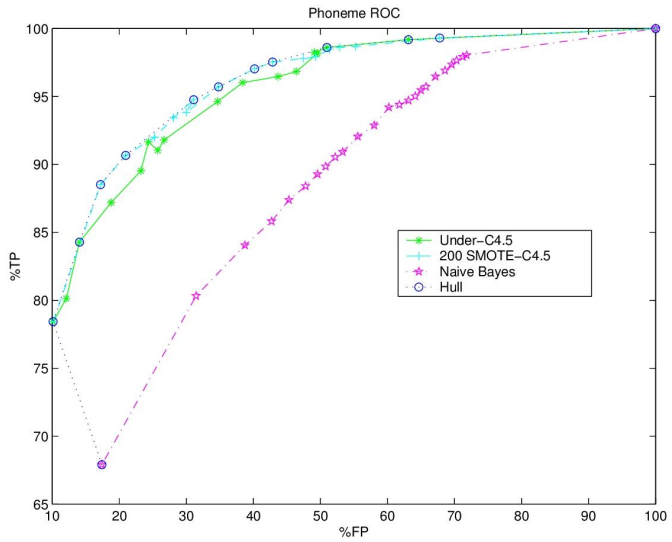


Figure 7: Phoneme. Comparison of SMOTE-C4.5, Under-C4.5, and Naive Bayes. SMOTE-C4.5 dominates over Naive Bayes and Under-C4.5 in the ROC space. SMOTE-C4.5 classifiers are potentially optimal classifiers.

Figures 9 through 23 show the experimental ROC curves obtained for the nine datasets with the three classifiers. The ROC curve for plain under-sampling of the majority class (Ling & Li, 1998; Japkowicz, 2000; Kubat & Matwin, 1997; Provost & Fawcett, 2001) is compared with our approach of combining synthetic minority class over-sampling (SMOTE) with majority class under-sampling. The plain under-sampling curve is labeled "Under", and the SMOTE and under-sampling combination ROC curve is labeled "SMOTE". Depending on the size and relative imbalance of the dataset, one to five SMOTE and under-sampling curves are created. We only show the best results from SMOTE combined with under-sampling and the plain under-sampling curve in the graphs. The SMOTE ROC curve from C4.5 is also compared with the ROC curve obtained from varying the priors of minority class using a Naive Bayes classifier — labeled as "Naive Bayes". "SMOTE", "Under", and "Loss Ratio" ROC curves, generated using Ripper are also compared. For a given family of ROC curves, an ROC convex hull (Provost & Fawcett, 2001) is generated. The ROC convex hull is generated using the Graham's algorithm (O'Rourke, 1998). For reference, we show the ROC curve that would be obtained using minority over-sampling by replication in Figure 19.

Each point on the ROC curve is the result of either a classifier (C4.5 or Ripper) learned for a particular combination of under-sampling and SMOTE, a classifier (C4.5 or Ripper) learned with plain under-sampling, or a classifier (Ripper) learned using some loss ratio or a classifier (Naive Bayes) learned for a different prior for the minority class. Each point represents the average (%TP and %FP) 10-fold cross-validation result. The lower leftmost point for a given ROC curve is from the raw dataset, without any majority class under-

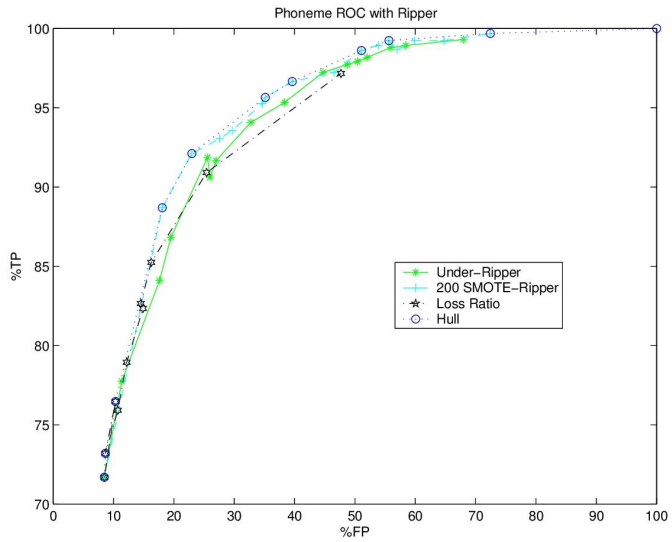


Figure 8: Phoneme. Comparison of SMOTE-Ripper, Under-Ripper, and modifying Loss Ratio in Ripper. SMOTE-Ripper dominates over Under-Ripper and Loss Ratio in the ROC space. More SMOTE-Ripper classifiers lie on the ROC convex hull.

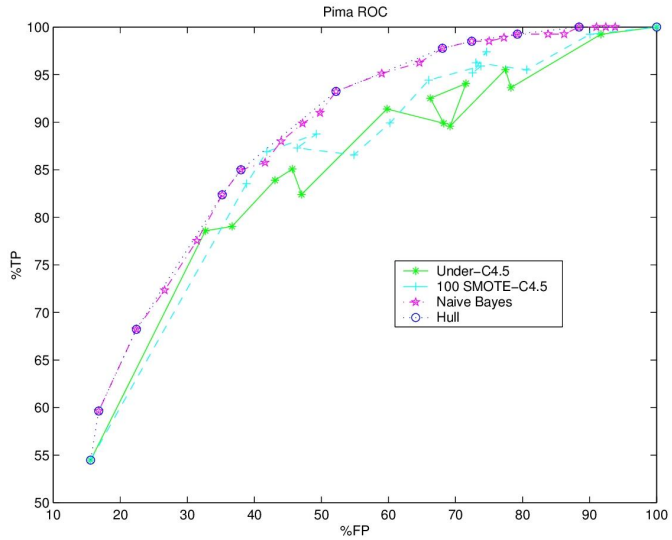


Figure 9: Pima Indians Diabetes. Comparison of SMOTE-C4.5, Under-C4.5, and Naive Bayes. Naive Bayes dominates over SMOTE-C4.5 in the ROC space.

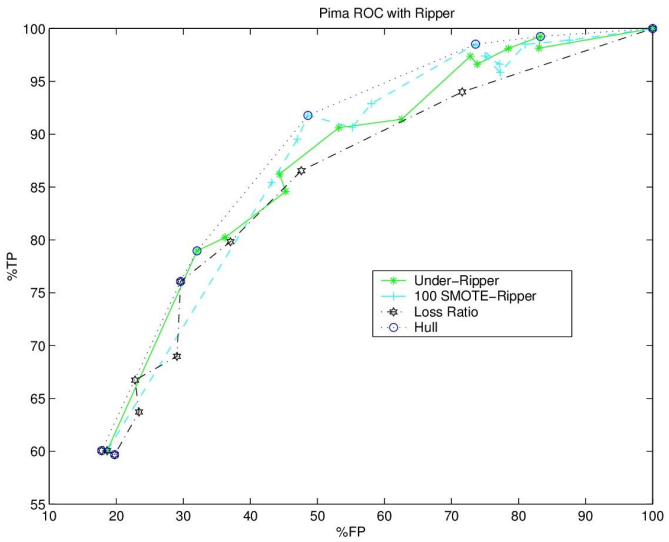


Figure 10: Pima Indians Diabetes. Comparison of SMOTE-Ripper, Under-Ripper, and modifying Loss Ratio in Ripper. SMOTE-Ripper dominates over Under-Ripper and Loss Ratio in the ROC space.

sampling or minority class over-sampling. The minority class was over-sampled at 50%, 100%, 200%, 300%, 400%, 500%. The majority class was under-sampled at 10%, 15%, 25%, 50%, 75%, 100%, 125%, 150%, 175%, 200%, 300%, 400%, 500%, 600%, 700%, 800%, 1000%, and 2000%. The amount of majority class under-sampling and minority class over-sampling depended on the dataset size and class proportions. For instance, consider the ROC curves in Figure 17 for the mammography dataset. There are three curves — one for plain majority class under-sampling in which the range of under-sampling is varied between 5% and 2000% at different intervals, one for a combination of SMOTE and majority class under-sampling, and one for Naive Bayes — and one ROC convex hull curve. The ROC curve shown in Figure 17 is for the minority class over-sampled at 400%. Each point on the SMOTE ROC curves represents a combination of (synthetic) over-sampling and under-sampling, the amount of under-sampling follows the same range as for plain under-sampling. For a better understanding of the ROC graphs, we have shown different sets of ROC curves for one of our datasets in Appendix A.

For the Can dataset, we had to SMOTE to a lesser degree than for the other datasets due to the structural nature of the dataset. For the Can dataset there is a structural neighborhood already established in the mesh geometry, so SMOTE can lead to creating neighbors which are under the surface (and hence not interesting), since we are looking at the feature space of physics variables and not the structural information.

The ROC curves show a trend that as we increase the amount of under-sampling coupled with over-sampling, our minority classification accuracy increases, of course at the expense of more majority class errors. For almost all the ROC curves, the SMOTE approach dom-

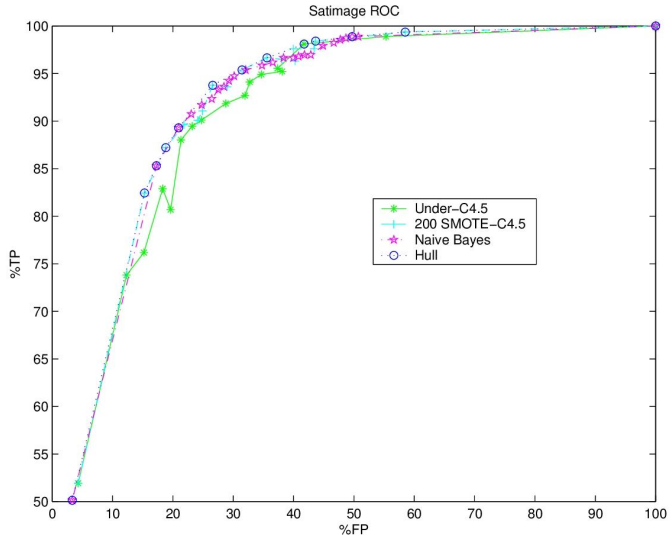


Figure 11: Satimage. Comparison of SMOTE-C4.5, Under-C4.5, and Naive Bayes. The ROC curves of Naive Bayes and SMOTE-C4.5 show an overlap; however, at higher TP's more points from SMOTE-C4.5 lie on the ROC convex hull.

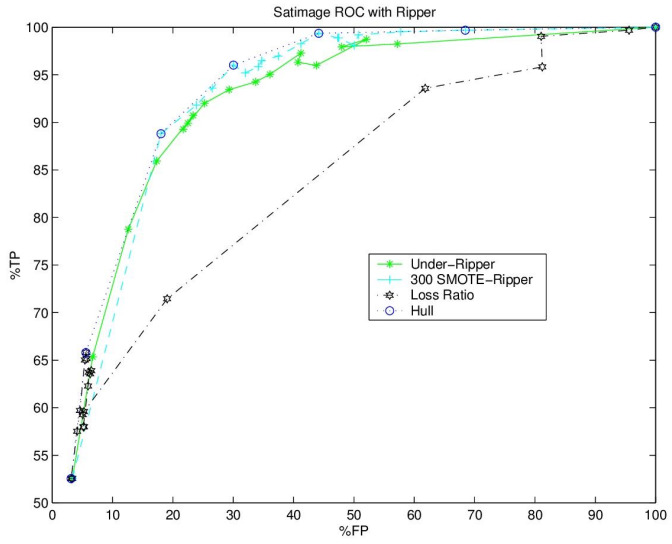


Figure 12: Satimage. Comparison of SMOTE-Ripper, Under-Ripper, and modifying Loss Ratio in Ripper. SMOTE-Ripper dominates the ROC space. The ROC convex hull is mostly constructed with points from SMOTE-Ripper.

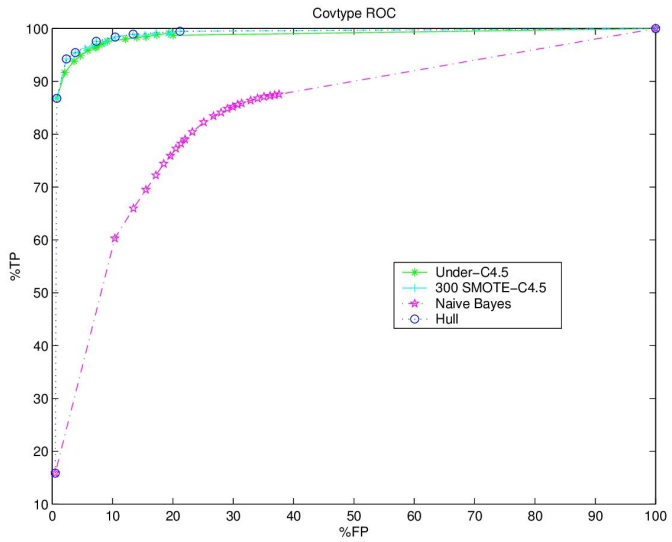


Figure 13: Forest Cover. Comparison of SMOTE-C4.5, Under-C4.5, and Naive Bayes. SMOTE-C4.5 and Under-C4.5 ROC curves are very close to each other. However, more points from the SMOTE-C4.5 ROC curve lie on the ROC convex hull, thus establishing a dominance.

inates. Adhering to the definition of ROC convex hull, most of the potentially optimal classifiers are the ones generated with SMOTE.

5.3 AUC Calculation

The Area Under the ROC curve (AUC) is calculated using a form of the trapezoid rule. The lower leftmost point for a given ROC curve is a classifier's performance on the raw data. The upper rightmost point is always (100%, 100%). If the curve does not naturally end at this point, the point is added. This is necessary in order for the AUC's to be compared over the same range of %FP.

The AUCs listed in Table 5.3 show that for all datasets the combined synthetic minority over-sampling and majority over-sampling is able to improve over plain majority under-sampling with C4.5 as the base classifier. Thus, our SMOTE approach provides an improvement in correct classification of data in the underrepresented class. The same conclusion holds from an examination of the ROC convex hulls. Some of the entries are missing in the table, as SMOTE was not applied at the same amounts to all datasets. The amount of SMOTE was less for less skewed datasets. Also, we have not included AUC's for Ripper/Naive Bayes. The ROC convex hull identifies SMOTE classifiers to be potentially optimal as compared to plain under-sampling or other treatments of misclassification costs, generally. Exceptions are as follows: for the Pima dataset, Naive Bayes dominates over SMOTE-C4.5; for the Oil dataset, Under-Ripper dominates over SMOTE-Ripper. For the Can dataset, SMOTE-*classifier* (*classifier* = C4.5 or Ripper) and Under-*classifier* ROC

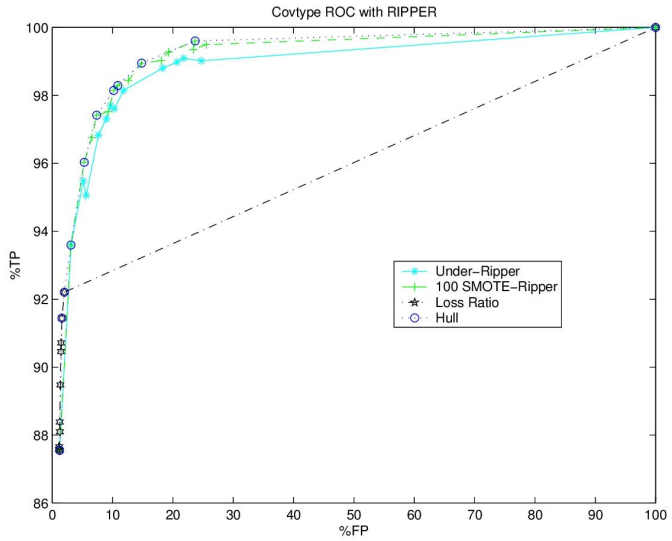


Figure 14: Forest Cover. Comparison of SMOTE-Ripper, Under-Ripper, and modifying Loss Ratio in Ripper. SMOTE-Ripper shows a domination in the ROC space. More points from SMOTE-Ripper curve lie on the ROC convex hull.

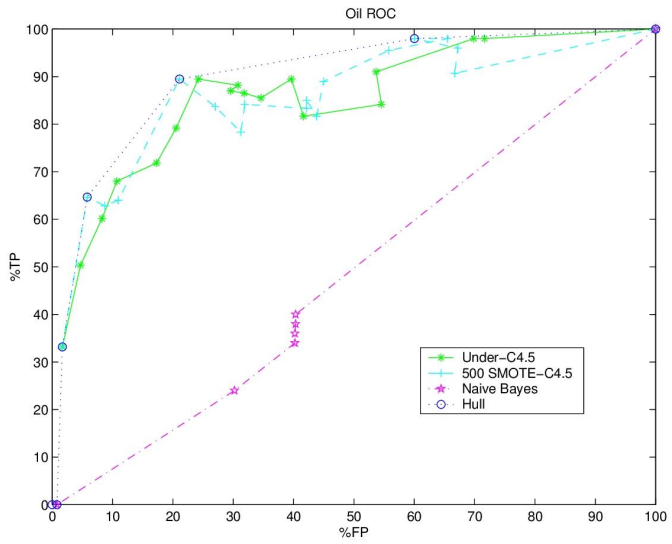


Figure 15: Oil. Comparison of SMOTE-C4.5, Under-C4.5, and Naive Bayes. Although, SMOTE-C4.5 and Under-C4.5 ROC curves intersect at points, more points from SMOTE-C4.5 curve lie on the ROC convex hull.

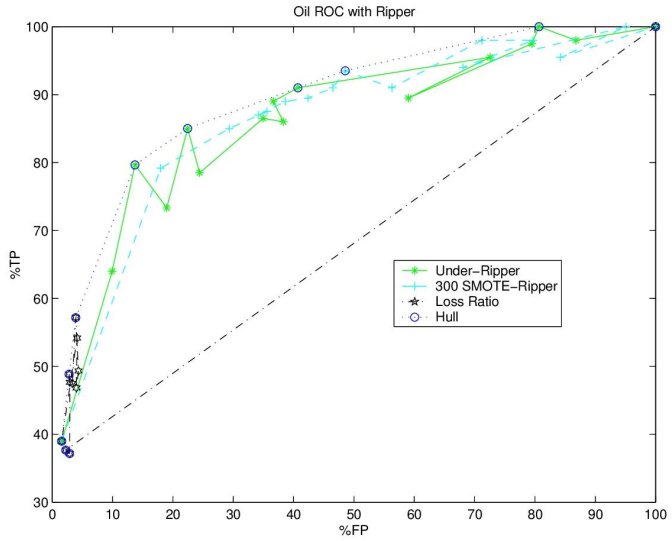


Figure 16: Oil. Comparison of SMOTE-Ripper, Under-Ripper, and modifying Loss Ratio in Ripper. Under-Ripper and SMOTE-Ripper curves intersect, and more points from the Under-Ripper curve lie on the ROC convex hull.

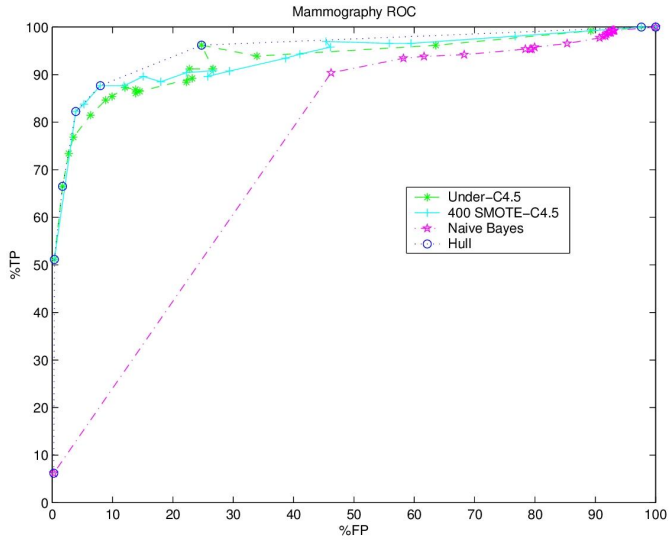


Figure 17: Mammography. Comparison of SMOTE-C4.5, Under-C4.5, and Naive Bayes. SMOTE-C4.5 and Under-C4.5 curves intersect in the ROC space; however, by virtue of number of points on the ROC convex hull, SMOTE-C4.5 has more potentially optimal classifiers.

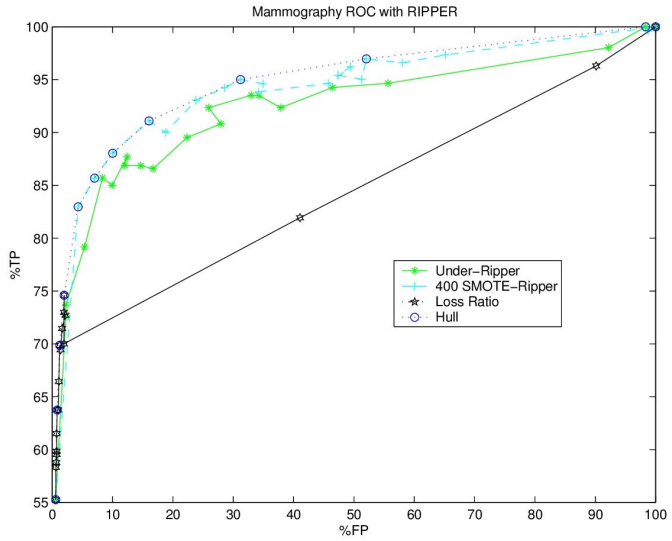


Figure 18: Mammography. Comparison of SMOTE-Ripper, Under-Ripper, and modifying Loss Ratio in Ripper. SMOTE-Ripper dominates the ROC space for TP > 75%.

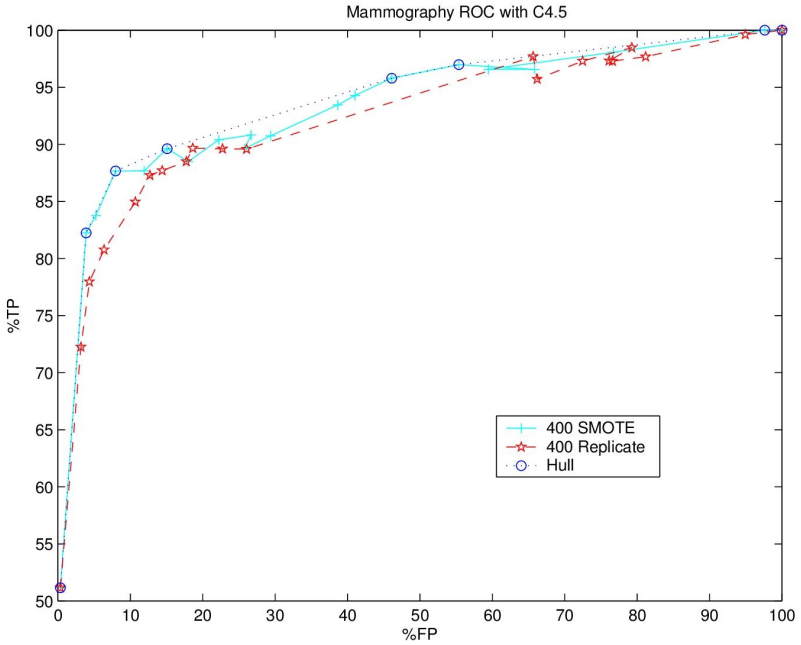


Figure 19: A comparison of over-sampling minority class examples by SMOTE and over-sampling the minority class examples by replication for the Mammography dataset.

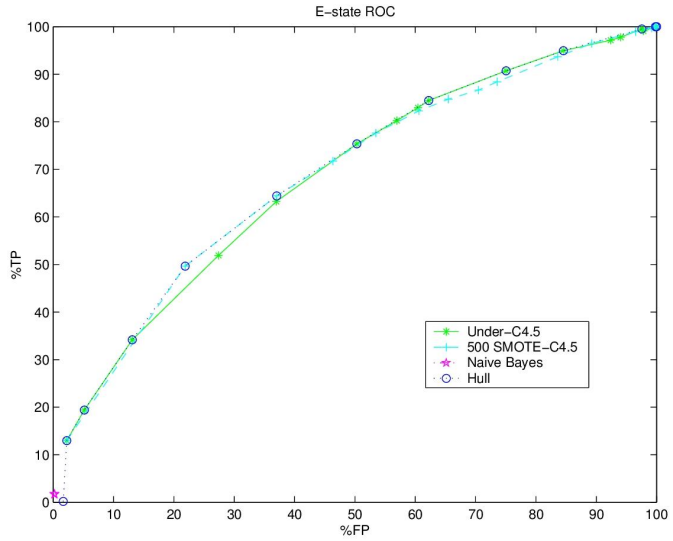


Figure 20: E-state. (a) Comparison of SMOTE-C4.5, Under-C4.5, and Naive Bayes. SMOTE-C4.5 and Under-C4.5 curves intersect in the ROC space; however, SMOTE-C4.5 has more potentially optimal classifiers, based on the number of points on the ROC convex hull.

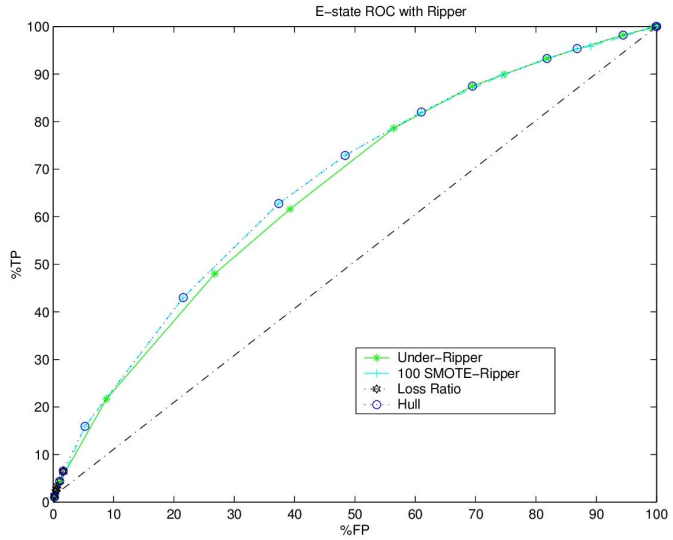


Figure 21: E-state. Comparison of SMOTE-Ripper, Under-Ripper, and modifying Loss Ratio in Ripper. SMOTE-Ripper has more potentially optimal classifiers, based on the number of points on the ROC convex hull.

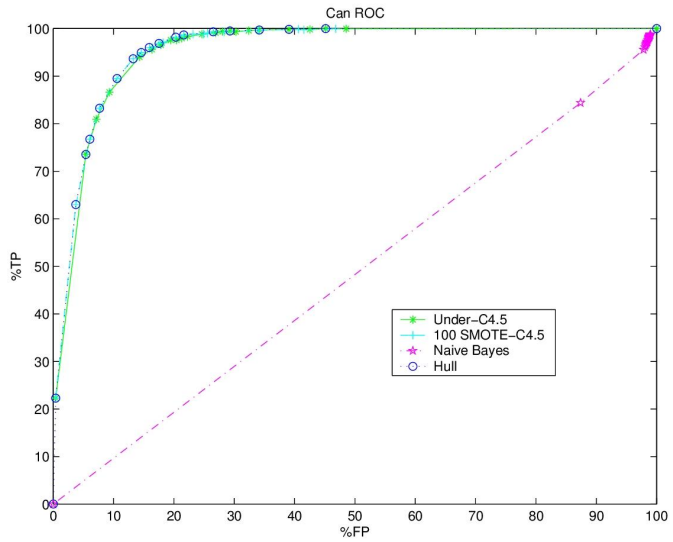


Figure 22: Can. Comparison of SMOTE-C4.5, Under-C4.5, and Naive Bayes. SMOTE-C4.5 and Under-C4.5 ROC curves overlap for most of the ROC space.

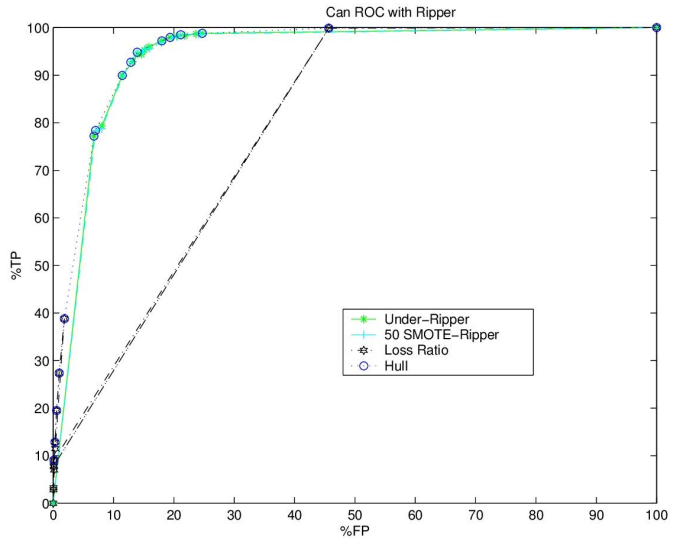


Figure 23: Can. Comparison of SMOTE-Ripper, Under-Ripper, and modifying Loss Ratio in Ripper. SMOTE-Ripper and Under-Ripper ROC curves overlap for most of the ROC space.

Dataset	Under	50 SMOTE	100 SMOTE	200 SMOTE	300 SMOTE	400 SMOTE	500 SMOTE
Pima	7242		**7307**				
Phoneme	8622		8644	**8661**			
Satimage	8900		8957	**8979**	8963	8975	8960
Forest Cover	9807		9832	9834	**9849**	9841	9842
Oil	8524		8523	8368	8161	8339	**8537**
Mammography	9260		9250	9265	9311	**9330**	9304
E-state	6811		6792	**6828**	6784	6788	6779
Can	9535	**9560**	9505	9505	9494	9472	9470

Table 3: AUC's [C4.5 as the base classifier] with the best highlighted in bold.

curves overlap in the ROC space. For all the other datasets, SMOTE-*classifier* has more potentially optimal classifiers than any other approach.

5.4 Additional comparison to changing the decision thresholds

Provost (2000) suggested that simply changing the decision threshold should always be considered as an alternative to more sophisticated approaches. In the case of C4.5, this would mean changing the decision threshold at the leaves of the decision trees. For example, a leaf could classify examples as the minority class even if more than 50% of the training examples at the leaf represent the majority class. We experimented by setting the decision thresholds at the leaves for the C4.5 decision tree learner at 0.5, 0.45, 0.42, 0.4, 0.35, 0.32, 0.3, 0.27, 0.25, 0.22, 0.2, 0.17, 0.15, 0.12, 0.1, 0.05, 0.0. We experimented on the Phoneme dataset. Figure 24 shows the comparison of the SMOTE and under-sampling combination against C4.5 learning by tuning the bias towards the minority class. The graph shows that the SMOTE and under-sampling combination ROC curve is dominating over the entire range of values.

5.5 Additional comparison to one-sided selection and SHRINK

For the oil dataset, we also followed a slightly different line of experiments to obtain results comparable to (Kubat et al., 1998). To alleviate the problem of imbalanced datasets the authors have proposed (a) one-sided selection for under-sampling the majority class (Kubat & Matwin, 1997) and (b) the SHRINK system (Kubat et al., 1998). Table 5.5 contains the results from (Kubat et al., 1998). Acc+ is the accuracy on positive (minority) examples and Acc− is the accuracy on the negative (majority) examples. Figure 25 shows the trend for Acc+ and Acc− for one combination of the SMOTE strategy and varying degrees of under-sampling of the majority class. The Y-axis represents the accuracy and the X-axis represents the percentage majority class under-sampled. The graphs indicate that in the band of under-sampling between 50% and 125% the results are comparable to those achieved by SHRINK and better than SHRINK in some cases. Table 5.5 summarizes the results for the SMOTE at 500% and under-sampling combination. We also tried combinations of SMOTE at 100-400% and varying degrees of under-sampling and achieved comparable results. The

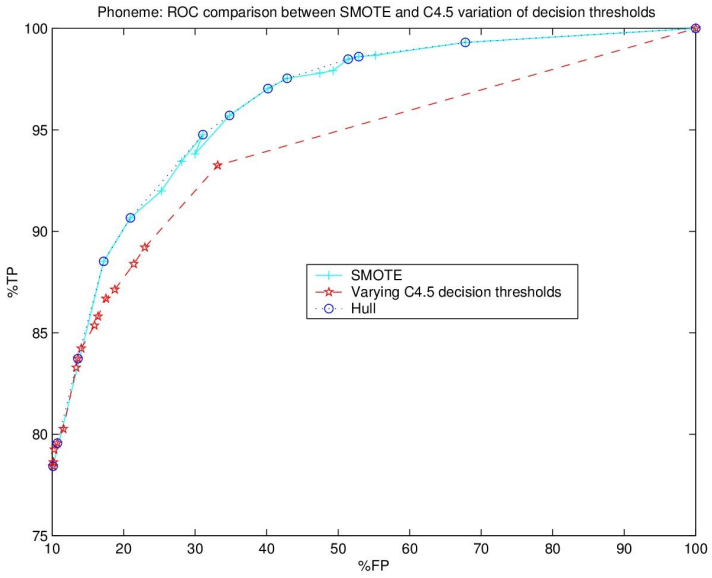


Figure 24: SMOTE and Under-sampling combination against C4.5 learning by tuning the bias towards the minority class

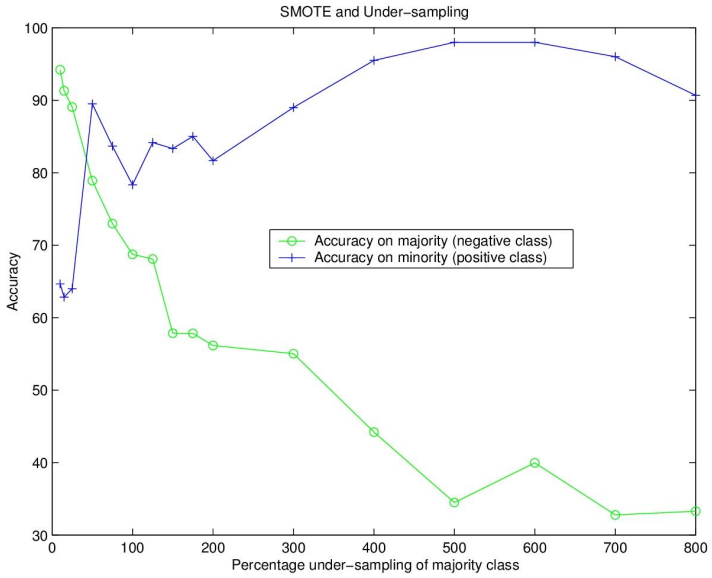


Figure 25: SMOTE (500 OU) and Under-sampling combination performance

SHRINK approach and our SMOTE approach are not directly comparable, though, as they see different data points. SMOTE offers no clear improvement over one-sided selection.

Method	Acc+	Acc−
SHRINK	82.5%	60.9%
One-sided selection	76.0%	86.6%

Table 4: Cross-validation results (Kubat et al., 1998)

Under-sampling %	Acc+	Acc−
10%	64.7%	94.2%
15%	62.8%	91.3%
25%	64.0%	89.1%
50%	89.5%	78.9%
75%	83.7%	73.0%
100%	78.3%	68.7%
125%	84.2%	68.1%
150%	83.3%	57.8%
175%	85.0%	57.8%
200%	81.7%	56.7%
300%	89.0%	55.0%
400%	95.5%	44.2%
500%	98.0%	35.5%
600%	98.0%	40.0%
700%	96.0%	32.8%
800%	90.7%	33.3%

Table 5: Cross-validation results for SMOTE at 500% SMOTE on the Oil data set.

6. Future Work

There are several topics to be considered further in this line of research. Automated adaptive selection of the number of nearest neighbors would be valuable. Different strategies for creating the synthetic neighbors may be able to improve the performance. Also, selecting nearest neighbors with a focus on examples that are incorrectly classified may improve performance. A minority class sample could possibly have a majority class sample as its nearest neighbor rather than a minority class sample. This crowding will likely contribute to the redrawing of the decision surfaces in favor of the minority class. In addition to these topics, the following subsections discuss two possible extensions of SMOTE, and an application of SMOTE to information retrieval.

6.1 SMOTE-NC

While our SMOTE approach currently does not handle data sets with all nominal features, it was generalized to handle mixed datasets of continuous and nominal features. We call this approach Synthetic Minority Over-sampling TEchnique-Nominal Continuous [SMOTE-NC]. We tested this approach on the Adult dataset from the UCI repository. The SMOTE-NC algorithm is described below.

1. Median computation: Compute the median of standard deviations of all continuous features for the minority class. If the nominal features differ between a sample and its potential nearest neighbors, then this median is included in the Euclidean distance computation. We use median to penalize the difference of nominal features by an amount that is related to the typical difference in continuous feature values.

2. Nearest neighbor computation: Compute the Euclidean distance between the feature vector for which k-nearest neighbors are being identified (minority class sample) and the other feature vectors (minority class samples) using the continuous feature space. For every differing nominal feature between the considered feature vector and its potential nearest-neighbor, include the median of the standard deviations previously computed, in the Euclidean distance computation. Table 2 demonstrates an example.

F1 = 1 2 3 A B C [Let this be the sample for which we are computing nearest neighbors]
F2 = 4 6 5 A D E
F3 = 3 5 6 A B K
So, Euclidean Distance between F2 and F1 would be:
Eucl = sqrt[(4-1)^2 + (6-2)^2 + (5-3)^2 + Med^2 + Med^2]
Med is the median of the standard deviations of continuous features of the minority class.
The median term is included twice for feature numbers 5: B→D and 6: C→E, which differ for the two feature vectors: F1 and F2.

Table 6: Example of nearest neighbor computation for SMOTE-NC.

3. Populate the synthetic sample: The continuous features of the new synthetic minority class sample are created using the same approach of SMOTE as described earlier. The nominal feature is given the value occuring in the majority of the k-nearest neighbors.

The SMOTE-NC experiments reported here are set up the same as those with SMOTE, except for the fact that we examine one dataset only. SMOTE-NC with the Adult dataset differs from our typical result: it performs worse than plain under-sampling based on AUC, as shown in Figures 26 and 27. We extracted only continuous features to separate the effect of SMOTE and SMOTE-NC on this dataset, and to determine whether this oddity was due to our handling of nominal features. As shown in Figure 28, even SMOTE with only continuous features applied to the Adult dataset, does not achieve any better performance than plain under-sampling. Some of the minority class continuous features have a very high variance, so, the synthetic generation of minority class samples could be overlapping with the majority class space, thus leading to more false positives than plain under-sampling. This hypothesis is also supported by the decreased AUC measure as we SMOTE at degrees greater than 50%. The higher degrees of SMOTE lead to more minority class samples in the dataset, and thus a greater overlap with the majority class decision space.

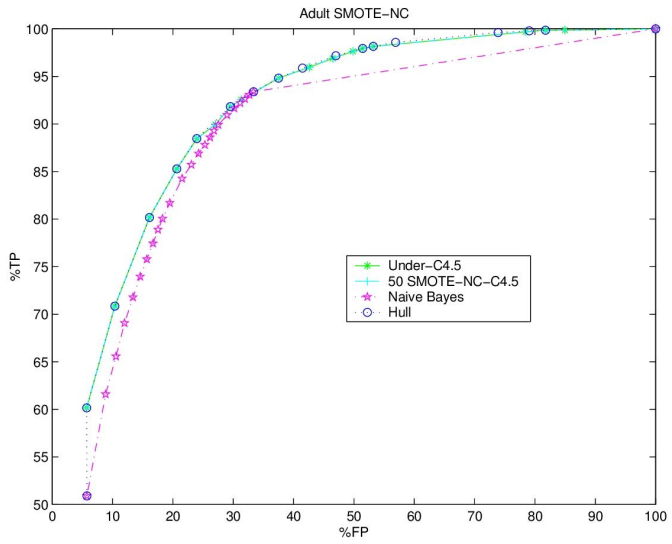


Figure 26: Adult. Comparison of SMOTE-C4.5, Under-C4.5, and Naive Bayes. SMOTE-C4.5 and Under-C4.5 ROC curves overlap for most of the ROC space.

6.2 SMOTE-N

Potentially, SMOTE can also be extended for nominal features — SMOTE-N — with the nearest neighbors computed using the modified version of Value Difference Metric (Stanfill & Waltz, 1986) proposed by Cost and Salzberg (1993). The Value Difference Metric (VDM) looks at the overlap of feature values over all feature vectors. A matrix defining the distance

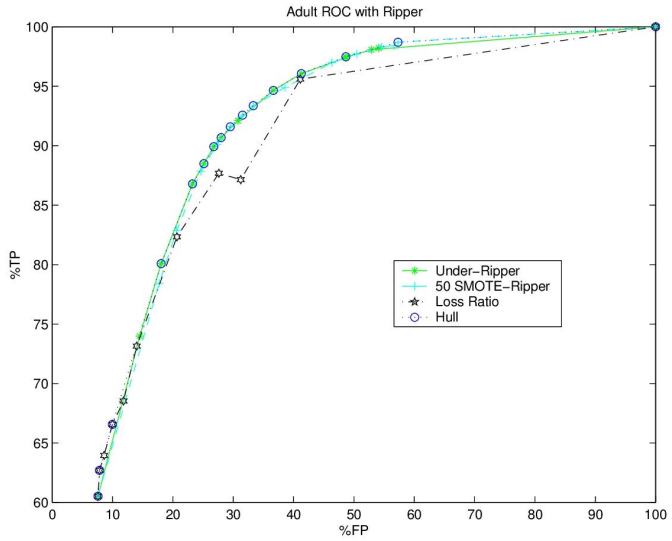


Figure 27: Adult. Comparison of SMOTE-Ripper, Under-Ripper, and modifying Loss Ratio in Ripper. SMOTE-Ripper and Under-Ripper ROC curves overlap for most of the ROC space.

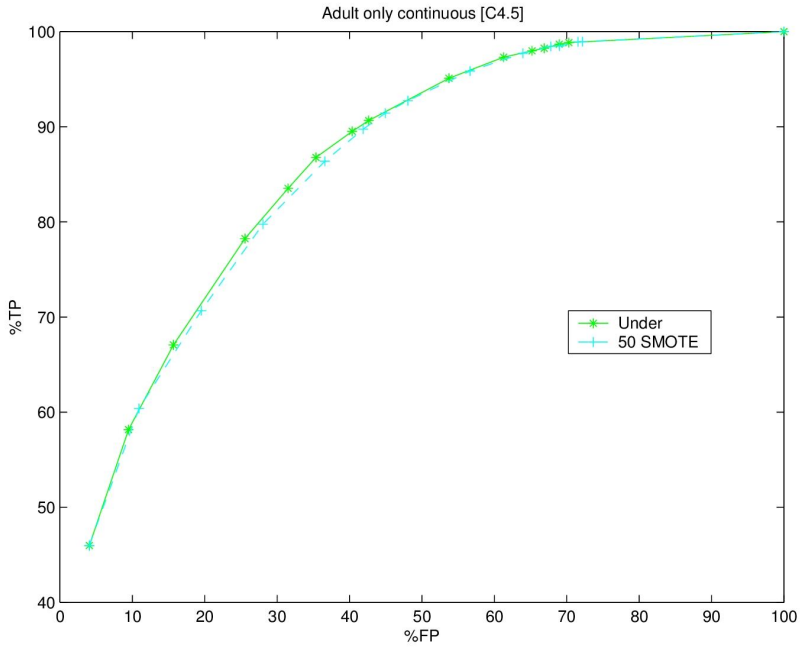


Figure 28: Adult with only continuous features. The overlap of SMOTE-C4.5 and Under-C4.5 is observed under this scenario as well.

between corresponding feature values for all feature vectors is created. The distance δ between two corresponding feature values is defined as follows.

$$\delta(V_1, V_2) = \sum_{i=1}^{n} |\frac{C_{1i}}{C_1} - \frac{C_{2i}}{C_2}|^k \tag{1}$$

In the above equation, V_1 and V_2 are the two corresponding feature values. C_1 is the total number of occurrences of feature value V_1, and C_{1i} is the number of occurrences of feature value V_1 for class i. A similar convention can also be applied to C_{2i} and C_2. k is a constant, usually set to 1. This equation is used to compute the matrix of value differences for each nominal feature in the given set of feature vectors. Equation 1 gives a geometric distance on a fixed, finite set of values (Cost & Salzberg, 1993). Cost and Salzberg's modified VDM omits the weight term w_f^a included in the δ computation by Stanfill and Waltz, which has an effect of making δ symmetric. The distance Δ between two feature vectors is given by:

$$\Delta(X, Y) = w_x w_y \sum_{i=1}^{N} \delta(x_i, y_i)^r \tag{2}$$

$r = 1$ yields the Manhattan distance, and $r = 2$ yields the Euclidean distance (Cost & Salzberg, 1993). w_x and w_y are the exemplar weights in the modified VDM. $w_y = 1$ for a new example (feature vector), and w_x is the bias towards more reliable examples (feature vectors) and is computed as the ratio of the number of uses of a feature vector to the number of correct uses of the feature vector; thus, more accurate feature vectors will have $w_x \approx 1$. For SMOTE-N we can ignore these weights in equation 2, as SMOTE-N is not used for classification purposes directly. However, we can redefine these weights to give more weight to the minority class feature vectors falling closer to the majority class feature vectors; thus, making those minority class features appear further away from the feature vector under consideration. Since, we are more interested in forming broader but accurate regions of the minority class, the weights might be used to avoid populating along neighbors which fall closer to the majority class. To generate new minority class feature vectors, we can create new set feature values by taking the majority vote of the feature vector in consideration and its k nearest neighbors. Table 6.2 shows an example of creating a synthetic feature vector.

Let F1 = A B C D E be the feature vector under consideration and let its 2 nearest neighbors be
F2 = A F C G N
F3 = H B C D N
The application of SMOTE-N would create the following feature vector:
FS = A B C D N

Table 7: Example of SMOTE-N

6.3 Application of SMOTE to Information Retrieval

We are investigating the application of SMOTE to information retrieval (IR). The IR problems come with a plethora of features and potentially many categories. SMOTE would have to be applied in conjunction with a feature selection algorithm, after transforming the given document or web page in a bag-of-words format.

An interesting comparison to SMOTE would be the combination of Naive Bayes and *Odds ratio*. *Odds ratio* focuses on a target class, and ranks documents according to their relevance to the target or positive class. SMOTE also focuses on a target class by creating more examples of that class.

7. Summary

The results show that the SMOTE approach can improve the accuracy of classifiers for a minority class. SMOTE provides a new approach to over-sampling. The combination of SMOTE and under-sampling performs better than plain under-sampling. SMOTE was tested on a variety of datasets, with varying degrees of imbalance and varying amounts of data in the training set, thus providing a diverse testbed. The combination of SMOTE and under-sampling also performs better, based on domination in the ROC space, than varying loss ratios in Ripper or by varying the class priors in Naive Bayes Classifier: the methods that could directly handle the skewed class distribution. SMOTE forces focused learning and introduces a bias towards the minority class. Only for Pima — the least skewed dataset — does the Naive Bayes Classifier perform better than SMOTE-C4.5. Also, only for the Oil dataset does the Under-Ripper perform better than SMOTE-Ripper. For the Can dataset, SMOTE-*classifier* and Under-*classifier* ROC curves overlap in the ROC space. For all the rest of the datasets SMOTE-*classifier* performs better than Under-*classifier*, Loss Ratio, and Naive Bayes. Out of a total of 48 experiments performed, SMOTE-*classifier* does not perform the best only for 4 experiments.

The interpretation of why synthetic minority over-sampling improves performance where as minority over-sampling with replacement does not is fairly straightforward. Consider the effect on the decision regions in feature space when minority over-sampling is done by replication (sampling with replacement) versus the introduction of synthetic examples. With replication, the decision region that results in a classification decision for the minority class can actually become smaller and more specific as the minority samples in the region are replicated. This is the opposite of the desired effect. Our method of synthetic over-sampling works to cause the classifier to build larger decision regions that contain nearby minority class points. The same reasons may be applicable to why SMOTE performs better than Ripper's loss ratio and Naive Bayes; these methods, nonetheless, are still learning from the information provided in the dataset, albeit with different cost information. SMOTE provides more related minority class samples to learn from, thus allowing a learner to carve broader decision regions, leading to more coverage of the minority class.

Acknowledgments

This research was partially supported by the United States Department of Energy through the Sandia National Laboratories ASCI VIEWS Data Discovery Program, contract number

DE-AC04-76DO00789. We thank Robert Holte for providing the oil spill dataset used in their paper. We also thank Foster Provost for clarifying his method of using the Satimage dataset. We would also like to thank the anonymous reviewers for their various insightful comments and suggestions.

Appendix A. ROC graphs for Oil Dataset

The following figures show different sets of ROC curves for the oil dataset. Figure 29 (a) shows the ROC curves for the Oil dataset, as included in the main text; Figure 29(b) shows the ROC curves without the ROC convex hull; Figure 29(c) shows the two convex hulls, obtained with and without SMOTE. The ROC convex hull shown by dashed lines and stars in Figure 29(c), was computed by including Under-C4.5 and Naive Bayes in the family of ROC curves. The ROC convex hull shown by solid line and small circles in Figure 29(c) was computed by including 500 SMOTE-C4.5, Under-C4.5, and Naive Bayes in the family of ROC curves. The ROC convex hull with SMOTE dominates the ROC convex hull without SMOTE, hence SMOTE-C4.5 contributes more optimal classifiers.

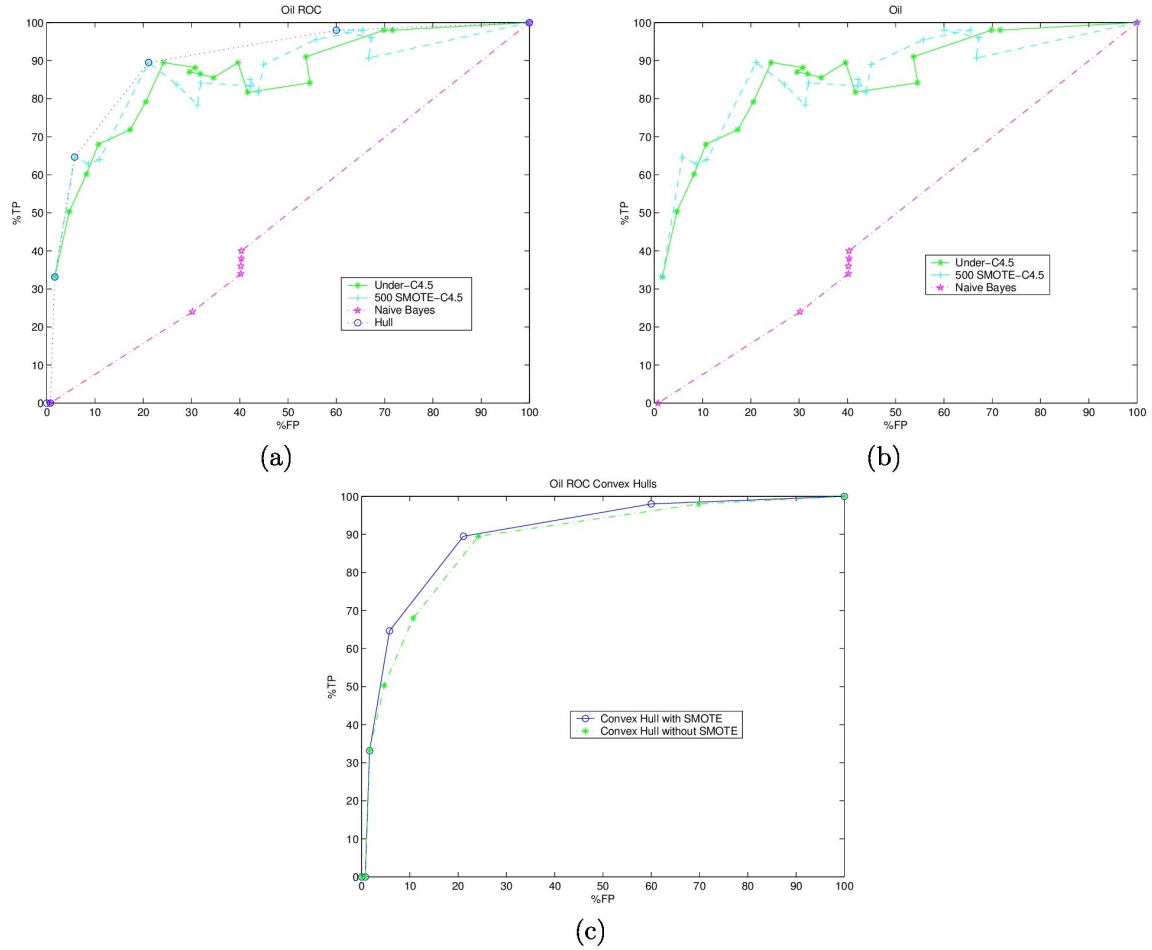


Figure 29: ROC curves for the Oil Dataset. (a) ROC curves for SMOTE-C4.5, Under-C4.5, Naive Bayes, and their ROC convex hull. (b) ROC curves for SMOTE-C4.5, Under-C4.5, and Naive Bayes. (c) ROC convex hulls with and without SMOTE.

References

Blake, C., & Merz, C. (1998). UCI Repository of Machine Learning Databases http://www.ics.uci.edu/~mlearn/~MLRepository.html. Department of Information and Computer Sciences, University of California, Irvine.

Bradley, A. P. (1997). The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms. *Pattern Recognition, 30(6)*, 1145–1159.

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, P. (2000). SMOTE: Synthetic Minority Over-sampling TEchnique. In *International Conference of Knowledge Based Computer Systems*, pp. 46–57. National Center for Software Technology, Mumbai, India, Allied Press.

Chawla, N., & Hall, L. (1999). Modifying MUSTAFA to capture salient data. Tech. rep. ISL-99-01, University of South Florida, Computer Science and Eng. Dept.

Cohen, W. (1995a). Learning to Classify English Text with ILP Methods. In *Proceedings of the 5th International Workshop on Inductive Logic Programming*, pp. 3–24. Department of Computer Science, Katholieke Universiteit Leuven.

Cohen, W. W. (1995b). Fast Effective Rule Induction. In *Proc. 12th International Conference on Machine Learning*, pp. 115–123 Lake Tahoe, CA. Morgan Kaufmann.

Cohen, W. W., & Singer, Y. (1996). Context-sensitive Learning Methods for Text Categorization. In Frei, H.-P., Harman, D., Schäuble, P., & Wilkinson, R. (Eds.), *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pp. 307–315 Zürich, CH. ACM Press, New York, US.

Cost, S., & Salzberg, S. (1993). A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning, 10*(1), 57–78.

DeRouin, E., Brown, J., Fausett, L., & Schneider, M. (1991). Neural Network Training on Unequally Represented Classes. In *Intellligent Engineering Systems Through Artificial Neural Networks*, pp. 135–141 New York. ASME Press.

Domingos, P. (1999). Metacost: A General Method for Making Classifiers Cost-sensitive. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 155–164 San Diego, CA. ACM Press.

Drummond, C., & Holte, R. (2000). Explicitly Representing Expected Cost: An Alternative to ROC Representation. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 198–207 Boston. ACM.

Duda, R., Hart, P., & Stork, D. (2001). *Pattern Classification.* Wiley-Interscience.

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive Learning Algorithms and Representations for Text Categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management.*, pp. 148–155.

Ezawa, K., J., Singh, M., & Norton, S., W. (1996). Learning Goal Oriented Bayesian Networks for Telecommunications Risk Management. In *Proceedings of the International Conference on Machine Learning, ICML-96*, pp. 139–147 Bari, Italy. Morgan Kauffman.

Fawcett, T., & Provost, F. (1996). Combining Data Mining and Machine Learning for Effective User Profile. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 8–13 Portland, OR. AAAI.

Ha, T. M., & Bunke, H. (1997). Off-line, Handwritten Numeral Recognition by Perturbation Method. *Pattern Analysis and Machine Intelligence, 19/5*, 535–539.

Hall, L., Mohney, B., & Kier, L. (1991). The Electrotopological State: Structure Information at the Atomic Level for Molecular Graphs. *Journal of Chemical Information and Computer Science, 31*(76).

Japkowicz, N. (2000). The Class Imbalance Problem: Significance and Strategies. In *Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI'2000): Special Track on Inductive Learning* Las Vegas, Nevada.

Kubat, M., Holte, R., & Matwin, S. (1998). Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning, 30*, 195–215.

Kubat, M., & Matwin, S. (1997). Addressing the Curse of Imbalanced Training Sets: One Sided Selection. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 179–186 Nashville, Tennesse. Morgan Kaufmann.

Lee, S. (2000). Noisy Replication in Skewed Binary Classification. *Computational Statistics and Data Analysis, 34*.

Lewis, D., & Catlett, J. (1994). Heterogeneous Uncertainity Sampling for Supervised Learning. In *Proceedings of the Eleventh International Conference of Machine Learning*, pp. 148–156 San Francisco, CA. Morgan Kaufmann.

Lewis, D., & Ringuette, M. (1994). A Comparison of Two Learning Algorithms for Text Categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pp. 81–93.

Ling, C., & Li, C. (1998). Data Mining for Direct Marketing Problems and Solutions. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)* New York, NY. AAAI Press.

Mladenić, D., & Grobelnik, M. (1999). Feature Selection for Unbalanced Class Distribution and Naive Bayes. In *Proceedings of the 16th International Conference on Machine Learning.*, pp. 258–267. Morgan Kaufmann.

O'Rourke, J. (1998). *Computational Geometry in C.* Cambridge University Press, UK.

Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., & Brunk, C. (1994). Reducing Misclassification Costs. In *Proceedings of the Eleventh International Conference on Machine Learning* San Francisco, CA. Morgan Kauffmann.

Provost, F., & Fawcett, T. (2001). Robust Classification for Imprecise Environments. *Machine Learning, 42/3*, 203–231.

Provost, F., Fawcett, T., & Kohavi, R. (1998). The Case Against Accuracy Estimation for Comparing Induction Algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 445–453 Madison, WI. Morgan Kauffmann.

Quinlan, J. (1992). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Solberg, A., & Solberg, R. (1996). A Large-Scale Evaluation of Features for Automatic Detection of Oil Spills in ERS SAR Images. In *International Geoscience and Remote Sensing Symposium*, pp. 1484–1486 Lincoln, NE.

Stanfill, C., & Waltz, D. (1986). Toward Memory-based Reasoning. *Communications of the ACM, 29*(12), 1213–1228.

Swets, J. (1988). Measuring the Accuracy of Diagnostic Systems. *Science, 240*, 1285–1293.

Tomek, I. (1976). Two Modifications of CNN. *IEEE Transactions on Systems, Man and Cybernetics, 6*, 769–772.

Turney, P. (1996). Cost Sensitive Bibliography. http://ai.iit.nrc.ca/bibiliographies/cost-sensitive.html.

van Rijsbergen, C., Harper, D., & Porter, M. (1981). The Selection of Good Search Terms. *Information Processing and Management, 17*, 77–91.

Woods, K., Doss, C., Bowyer, K., Solka, J., Priebe, C., & Kegelmeyer, P. (1993). Comparative Evaluation of Pattern Recognition Techniques for Detection of Microcalcifications in Mammography. *International Journal of Pattern Recognition and Artificial Intelligence, 7(6)*, 1417–1436.

.

Appendix B : Borderline-SMOTE

Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning

Hui Han[1], Wen-Yuan Wang[1], and Bing-Huan Mao[2]

[1] Department of Automation, Tsinghua University, Beijing 100084, P. R. China
hanh01@mails.tsinghua.edu.cn
wwy-dau@mail.tsinghua.edu.cn
[2] Department of Statistics, Central University of Finance and Economics, Beijing 100081, P. R. China
maobinghuan@yahoo.com

Abstract. In recent years, mining with imbalanced data sets receives more and more attentions in both theoretical and practical aspects. This paper introduces the importance of imbalanced data sets and their broad application domains in data mining, and then summarizes the evaluation metrics and the existing methods to evaluate and solve the imbalance problem. Synthetic minority over-sampling technique (SMOTE) is one of the over-sampling methods addressing this problem. Based on SMOTE method, this paper presents two new minority over-sampling methods, borderline-SMOTE1 and borderline-SMOTE2, in which only the minority examples near the borderline are over-sampled. For the minority class, experiments show that our approaches achieve better TP rate and F-value than SMOTE and random over-sampling methods.

1 Introduction

There may be two kinds of imbalances in a data set. One is between-class imbalance, in which case some classes have much more examples than others [1]. The other is within-class imbalance, in which case some subsets of one class have much fewer examples than other subsets of the same class [2]. By convention, in imbalanced data sets, we call the classes having more examples the majority classes and the ones having fewer examples the minority classes.

The problem of imbalance has got more and more emphasis in recent years. Imbalanced data sets exists in many real-world domains, such as spotting unreliable telecommunication customers [3], detection of oil spills in satellite radar images [4], learning word pronunciations [5], text classification [6], detection of fraudulent telephone calls [7], information retrieval and filtering tasks [8], and so on. In these domains, what we are really interested in is the minority class other than the majority class. Thus, we need a fairly high prediction for the minority class. However, the traditional data mining algorithms behaves undesirable in the instance of imbalanced data sets, as the distribution of the data sets is not taken into consideration when these algorithms are designed.

The structure of this paper is organized as follows. Section 2 gives a brief introduction to the recent developments in the domains of imbalanced data sets. Section 3

D.S. Huang, X.-P. Zhang, G.-B. Huang (Eds.): ICIC 2005, Part I, LNCS 3644, pp. 878 – 887, 2005.

describes our over-sampling methods on resolving the imbalanced problem. Section 4 presents the experiments and compares our methods with other over-sampling methods. Section 5 draws the conclusion.

2. The Recent Developments in Imbalanced Data Sets Learning

2.1 Evaluation Metrics in Imbalanced Domains

Most of the studies in imbalanced domains mainly concentrate on two-class problem as multi-class problem can be simplified to two-class problem. By convention, the class label of the minority class is positive, and the class label of the majority class is negative. Table 1 illustrates a confusion matrix of a two-class problem. The first column of the table is the actual class label of the examples, and the first row presents their predicted class label. TP and TN denote the number of positive and negative examples that are classified correctly, while FN and FP denote the number of misclassified positive and negative examples respectively.

Table 1. Confusion matrix for a two-class problem

	Predicted Positive	Predicted Negative
Positive	TP	FN
Negative	FP	TN

$$\text{Accuracy} = (\text{TP}+\text{TN})/(\text{TP}+\text{FN}+\text{FP}+\text{TN}) \tag{1}$$

$$\text{FP rate} = \text{FP}/(\text{TN}+\text{FP}) \tag{2}$$

$$\text{TP rate} = \text{Recall} = \text{TP}/(\text{TP}+\text{FN}) \tag{3}$$

$$\text{Precision} = \text{TP}/(\text{TP}+\text{FP}) \tag{4}$$

$$F-value = ((1+\beta^2)\cdot \text{Re}call \cdot \text{Pr}ecision)/(\beta^2 \cdot \text{Re}call + \text{Pr}ecision) \tag{5}$$

When used to evaluate the performance of a learner for imbalanced data sets, accuracy is generally apt to predict the majority class better and behaves poorly to the minority class. We can come to this conclusion from its definition (formula (1)): if the dataset is extremely imbalanced, even when the classifier classifies all the majority examples correctly and misclassifies all the minority examples, the accuracy of the learner is still high because there are much more majority examples than minority examples. Under the circumstance, accuracy can not reflect reliable prediction for the minority class. Thus, more reasonable evaluation metrics are needed.

ROC curve [9] is one of the popular metrics to evaluate the learners for imbalanced data sets. It is a two-dimensional graph in which TP rate is plotted on the y-axis and

FP rate is plotted on the x-axis. FP rate (formula (2)) denotes the percentage of the misclassified negative examples, and TP rate (formula (3)) is the percentage of the correctly classified positive examples. The point (0, 1) is the ideal point of the learners. ROC curve depicts relative trade-offs between benefits (TP rate) and costs (FP rate). AUC (Area under ROC) can also be applied to evaluate the imbalanced data sets [9]. Furthermore, F-value (formula (5)) is also a popular evaluation metric for imbalance problem [10]. It is a kind of combination of recall (formula (3)) and precision (formula (4)), which are effective metrics for information retrieval community where the imbalance problem exist. F-value is high when both recall and precision are high, and can be adjusted through changing the value of β, where β corresponds to relative importance of precision vs. recall and it is usually set to 1.

The above evaluation metrics can reasonably evaluate the learner for imbalanced data sets because their formulae are relative to the minority class.

2.2 Methods for Dealing with Imbalanced Data Sets Learning

The solutions to imbalanced data sets can be divided into data and algorithmic levels. categories. The methods at data level change the distribution of the imbalanced data sets, and then the balanced data sets are provided to the learner to improve the detection rate of minority class. The methods at the algorithm level modify the existing data mining algorithms or put forward new algorithms to resolve the imbalance problem.

2.2.1 The Methods at Data Level

At the data level, different forms of re-sampling methods were proposed [1]. The simplest re-sampling methods are random over-sampling and random under-sampling. The former augments the minority class by exactly duplicating the examples of the minority class, while the latter randomly takes away some examples of the majority class. However, random over-sampling may make the decision regions of the learner smaller and more specific, thus cause the learner to over-fit. Random under-sampling can reduce some useful information of the data sets. Many improved re-sampling methods are thus presented, such as heuristic re-sampling methods, combination of over-sampling and under-sampling methods, embedding re-sampling methods into data mining algorithms, and so on. Some of the improved re-sampling methods are as follows.

Kubat et al. presented a heuristic under-sampling method which balanced the data set through eliminating the noise and redundant examples of the majority class [11]. Nitesh et al. over-sampled the minority class through SMOTE (Synthetic Minority Over-sampling Technique) method, which generated new synthetic examples along the line between the minority examples and their selected nearest neighbors [12]. The advantage of SMOTE is that it makes the decision regions larger and less specific. Nitesh et al. integrated SMOTE into a standard boosting procedure, thus improved the prediction of the minority class while not sacrificing the accuracy of the whole testing set [13]. Gustavo et al. combined over-sampling and under-sampling methods to resolve the imbalanced problem [14]. Andrew Estabrooks et al. proposed a multiple re-sampling method which selected the most appropriate re-sampling rate adaptively [15]. Taeho Jo et al. put forward a cluster-based over-sampling method which dealt

with between-class imbalance and within-class imbalance simultaneously [16]. Hongyu Guo et al. found out hard examples of the majority and minority classes during the process of boosting, then generated new synthetic examples from hard examples and add them to the data sets [17].

2.2.2 The Methods at Algorithm Level

The methods at algorithm level operate on the algorithms other than the data sets. The standard boosting algorithm, e.g. Adaboost [18], increases the weights of misclassified examples and decreases those correctly classified using the same proportion, without considering the imbalance of the data sets. Thus, traditional boosting algorithms do not perform well on the minority class. Aiming at the disadvantage above, Mahesh V. Joshi et al. proposed an improved boosting algorithm which updated weights of positive prediction (TP and FP) differently from weights of negative prediction (TN and FN). The new algorithm can achieve better prediction for the minority class [19]. When dealing with imbalanced data sets, the class boundary learned by Support Vector Machines (SVMs) is apt to skew toward the minority class, thus increase the misclassified rate of the minority class. Gang Wu et al. proposed class-boundary alignment algorithm which modify the class boundary through changing the kernel function of SVMs [20]. Kaizhu Huang et al. presented Biased Minimax Probability Machine (BMPM) to resolve the imbalance problem. Given the reliable mean and covariance matrices of the majority and minority classes, BMPM can derive the decision hyperplane by adjusting the lower bound of the real accuracy of the testing set [21]. Furthermore, there are other effective methods such as cost-based learning, adjusting the probability of the learners and one-class learning, and so on [22] [23].

3 A New Over-Sampling Method: Borderline-SMOTE

In order to achieve better prediction, most of the classification algorithms attempt to learn the borderline of each class as exactly as possible in the training process. The examples on the borderline and the ones nearby (we call them borderline examples in this paper) are more apt to be misclassified than the ones far from the borderline, and thus more important for classification.

Based on the analysis above, those examples far from the borderline may contribute little to classification. We thus present two new minority over-sampling methods, borderline-SMOTE1 and borderline-SMOTE2, in which only the borderline examples of the minority class are over-sampled. Our methods are different from the existing over-sampling methods in which all the minority examples or a random subset of the minority class are over-sampled [1] [2] [12].

Our methods are based on SMOTE (Synthetic Minority Over-sampling Technique) [12]. SMOTE generates synthetic minority examples to over-sample the minority class. For every minority example, its k (which is set to 5 in SMOTE) nearest neighbors of the same class are calculated, then some examples are randomly selected from them according to the over-sampling rate. After that, new synthetic examples are generated along the line between the minority example and its selected nearest neighbors. Not like the existing over-sampling methods, our methods only over-sample or strengthen the borderline minority examples. First, we find out the border-

line minority examples; then, synthetic examples are generated from them and added to the original training set. Suppose that the whole training set is *T*, the minority class is *P* and the majority class is *N*, and

$$P = \{p_1, p_2, ..., p_{pnum}\}, N = \{n_1, n_2, ..., n_{nnum}\}$$

where *pnum* and *nnum* are the number of minority and majority examples. The detailed procedure of borderline-SMOTE1 is as follows.

Step 1. For every $p_i (i = 1,2,..., pnum)$ *in the minority class P, we calculate its m nearest neighbors from the whole training set T. The number of majority examples among the m nearest neighbors is denoted by* $m' (0 \le m' \le m)$.

Step 2. If $m' = m$, *i.e. all the m nearest neighbors of* p_i *are majority examples,* p_i *is considered to be noise and is not operated in the following steps. If* $m/2 \le m' < m$, *namely the number of* p_i*'s majority nearest neighbors is larger than the number of its minority ones,* p_i *is considered to be easily misclassified and put into a set DANGER. If* $0 \le m' < m/2$, p_i *is safe and needs not to participate in the follows steps.*

Step 3. The examples in DANGER are the borderline data of the minority class P, and we can see that $DANGER \subseteq P$. *We set*

$$DANGER = \{p'_1, p'_2, ..., p'_{dnum}\}, \quad 0 \le dnum \le pnum$$

For each example in *DANGER*, we calculate its *k* nearest neighbors from *P*.

Step 4. In this step, we generate $s \times dnum$ *synthetic positive examples from the data in DANGER, where s is an integer between 1 and k. For each* p'_i, *we randomly select s nearest neighbors from its k nearest neighbors in P. Firstly, we calculate the differences,* dif_j $(j = 1,2,..., s)$ *between* p'_i *and its s nearest neighbors from P, then multiply* dif_j *by a random number* r_j $(j = 1,2,..., s)$ *between 0 and 1, finally, s new synthetic minority examples are generated between* p'_i *and its nearest neighbors:*

$$synthetic_j = p'_i + r_j \times dif_j, \quad j = 1,2,..., s$$

We repeat the above procedure for each p'_i in *DANGER* and can attain $s \times dnum$ synthetic examples. This step is similar with SMOTE, for more detail see [12].

In the procedure above, p_i, n_i, p'_i, dif_j and $synthetic_j$ are vectors. We can see that new synthetic data are generated along the line between the minority borderline examples and their nearest neighbors of the same class, thus strengthened the borderline examples.

Borderline-SMOTE2 not only generates synthetic examples from each example in *DANGER* and its positive nearest neighbors in *P*, but also does that from its nearest negative neighbor in *N*. The difference between it and its nearest negative neighbor is

multiplied a random number between 0 and 0.5, thus the new generated examples are closer to the minority class.

Our methods can be easily understood with the following simulated data set, Circle, which has two classes. Fig. 1 (a) shows the original distribution of the data set, the circle points represent majority examples and the plus signs are minority examples. Firstly, we apply borderline-SMOTE to find out the borderline examples of the minority class, which are denoted by solid squares in Fig. 1 (b). Then, new synthetic examples are generated through those borderline examples of the minority class. The synthetic examples are shown in Fig. 1 (c) with hollow squares. It is easy to find out from the figures that, different from SMOTE, our methods only over-sample or strengthen the borderline and its nearby points of the minority class.

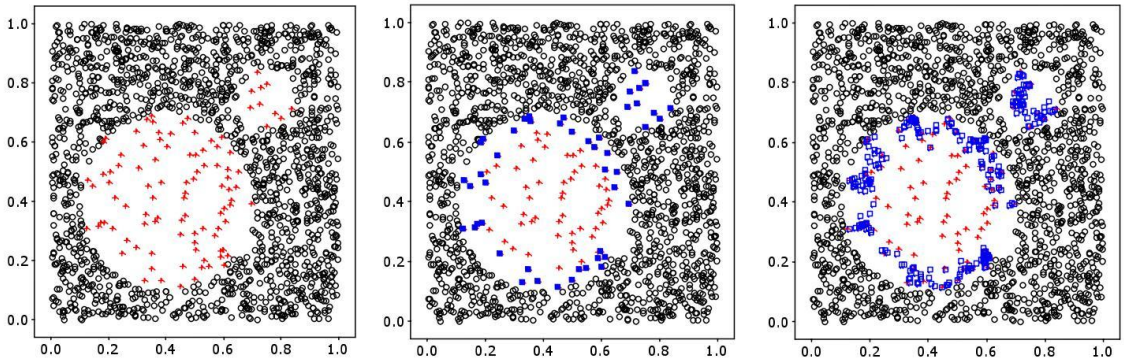


Fig. 1. (a) The original distribution of Circle data set. (b) The borderline minority examples (*solid squares*). (c) The borderline synthetic minority examples (*hollow squares*).

4 Experiments

We use TP rate and F-value for the minority class to evaluate the results of our experiments. TP rate denotes the accuracy of the minority class. And the value of β in F-value is set to 1 in this paper.

The four data sets used in our experiments are shown in Table 2. Among the four data sets, Circle is our simulated data set depicted in Fig. 1, and the others are from UCI [24]. All the attributes in the data sets are quantitative. For Satimage, we choose class label "4" as the minority class and regard the remainders as the majority class, as we only study two-class problem in this paper.

Table 2. The description of the data sets

The name of Data set	number of Examples	number of Attributes	Class label (minority : majority)	Percentage of minority class
Circle(Simulation)	1600	2	1:0	6.25%
Pima(UCI)	768	8	1:0	34.77%
Satimage(UCI)	6435	36	4:remainder	9.73%
Haberman(UCI)	306	3	Die : Survive	26.47%

In our experiments, four over-sampling methods are applied to the data sets: SMOTE, random over-sampling and our methods, borderline-SMOTE1 and borderline-SMOTE2, among which random over-sampling method augments the minority class by exactly duplicating the positive examples partly or completely [1]. Through increasing the number of examples in the minority class, over-sampling methods can balance the distribution of the data sets and improve the detection rate of the minority class.

In order to compare the results conveniently, the value of m in our methods is set in a way that, the number of the minority examples in *DANGER* is about half of the minority class. The value of k is set to 5 like SMOTE. For each method, the TP rates and F-values are attained through 10-fold cross-validation. In order to decrease the randomness in SMOTE and our methods, the TP rates and F-values for these methods are the average results of three independent 10-fold cross-validation experiments. After the original training sets are over-sampled with the methods above, C4.5 is applied as the validation classifier [25].

Since the nature of imbalance problem is to improve the prediction performance of the minority class, we only present the results of the minority class. We compare the results of the data sets through TP rate and F-value of the minority class. TP rate reflects the performance of the learner on the minority class of the testing set, while F-value shows the performance of the learner on the whole testing set.

Fig. 2 shows our experimental results. In the figure, (a), (b), (c) and (d) depict the F-value and TP rate for the minority class when the four over-sampling methods are applied on Circle, Pima, Satimage and Haberman respectively. The x-axis in each figure is the number of the new synthetic examples. The F-value and TP rate of the original data sets with C4.5 are also shown in the figures.

The results illustrated in Fig. 2 reveal the following results. First of all, all the four over-sampling methods improve TP rate of the minority class. For Circle, Pima and Haberman, the TP rates of our methods are better than SMOTE and random over-sampling. Comparing with the original data sets, the best improvements of TP rate for borderline-SMOTE1 and borderline-SMOTE2 on Circle are 20 and 22 per cent, 21.3 and 20.5 per cent on Pima, 10.1 and 10.0 per cent on Satimage, and both 45.2 per cent on Haberman. For Satimage, the TP rates of our methods are lager than that of random over-sampling, and are comparable with SMOTE. Secondly, the F-value of borderline-SMOTE1 is generally better than SMOTE and random over-sampling, and the F-value of borderline-SMOTE2 is also comparable with others. Comparing with the original data sets, the best improvements of F-value for borderline-SMOTE1 and borderline-SMOTE2 on Circle are 12.1 and 10.3 per cent, 2.3 and 1.3 per cent on Pima, 2.3 and 1.4 per cent on Satimage, and 24.7 and 23.0 per cent on Haberman.

As a whole, border-SMOTE1 behaves excellent on both TP rate and F-value, and borderline-SMOTE2 behaves super on TP rate because it generates synthetic examples from both the minority borderline examples and their nearest neighbors of the majority class, however, the procedure causes overlap between the two classes, thus decreases its F-value to some extent.

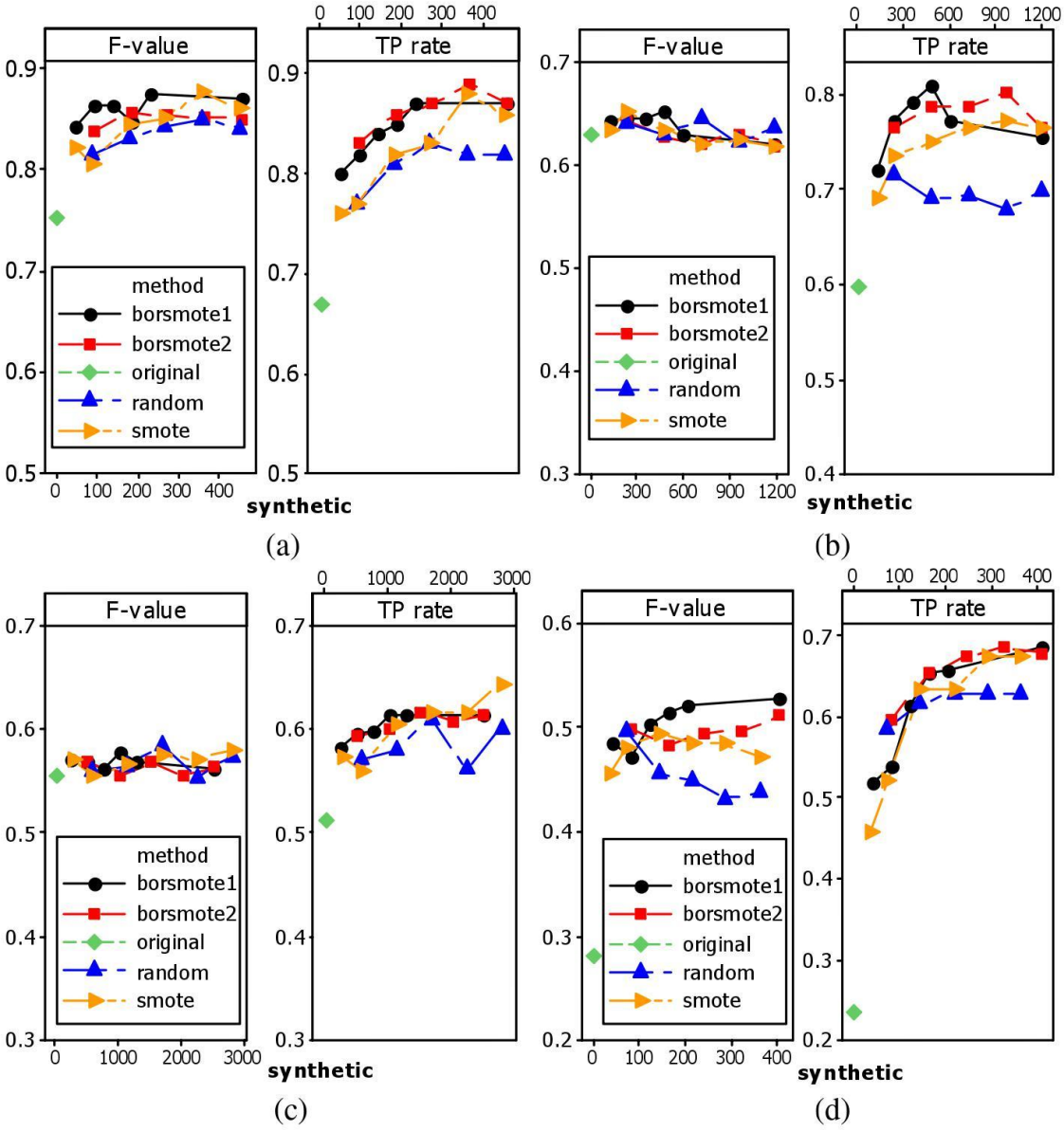


Fig. 2. (a), (b), (c) and (d) illustrate the F-value and TP rate for minority class when proposed over-sampling methods are applied on Circle, Pima, Satimge and Haberman respectively with C4.5. "borsmote1" and "borsmote2" denote borderline-SMOTE1 and borderline-SMOTE2, "random" denotes random over-sampling, and "original" denotes the values of the original data sets. The x-axis is the number of synthetic examples

5 Conclusion

In recent years, learning with imbalanced data sets receives more and more attentions in both theoretical and practical aspects. However, traditional data mining methods are not satisfactory. Aiming to solve the problem, two new synthetic minority over-sampling methods, borderline-SMOTE1 and borderline-SMOTE2 are presented in this paper. We compared the TP rate and F-value of our methods with SMOTE, random over-sampling and the original C4.5 for four data sets.

The borderline examples of the minority class are more easily misclassified than those ones far from the borderline. Thus our methods only over-sample the borderline examples of the minority class, while SMOTE and random over-sampling augment the minority class through all the examples from the minority class or a random subset of the minority class. Experiments indicate that our methods behave better, which validates the efficiency of our methods.

There are several topics left to be considered further in this line of research. Different strategies to define the *DANGER* examples, and automated adaptive determination of the number of examples in *DANGER* would be valuable. The combination of our methods with under-sampling methods and the integration of our methods to some data mining algorithms, are also worth trying.

References

1. Nitesh V.Chawla, Nathalie Japkowicz and Aleksander Kolcz.: Editorial: Special Issue on Learning from Imbalanced Data Sets. SIGKDD Explorations 6 (1) (2004) 1-6
2. G. Weiss: Mining with rarity: A unifying framework. SIGKDD Explorations 6 (1) (2004) 7-19
3. Ezawa, K.J., Singh, M. and Norton, S.W.: Learning Goal Oriented Bayesian Networks for Telecommunications Management. In Proceedings of the International Conference on Machine Learning, ICML'96(pp. 139-147), Bari, Italy, Morgan Kaufmann (1996)
4. Kubat, m., Holte, R., and Matwin, S.: Machine Learning for the Detection of Oil Spills in Satellite Radar Images. Machine Learning 30 195-215
5. A. van den Bosch, T. Weijters, H. J. van den Herik, and W. Daelemans: When small disjuncts abound, try lazy learning: A case study. In Proceedings of the Seventh Belgian-Dutch Conference on Machine Learning (1997) 109-118
6. Zhaohui Zheng, Xiaoyun Wu, Rohini Srihari: Feature Selection for Text Categorization on Imbalanced Data. SIGKDD Explorations 6 (1) (2004) 80-89
7. Fawcett, T.and Provost, F.: Combining Data Mining and Machine Learning for Effective User Profile. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland OR, AAAI Press (1996) 8-13
8. Lewis, D. and Catlett, Heterogeneous, J.: Uncertainty Sampling for Supervized Learning. Proceedings of the 11th International Conference on Machine Learning, ICML'94 (1994) 148-156
9. Bradley A.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30 (7) (1997) 1145-1159
10. Rijsbergen, C. J. van: Information Retrieval, Butterworths, London (1979)
11. Kubat, M., and Matwin, S. Addressing the Course of Imbalanced Training Sets: One-sided Selection. In ICML'97 (1997) 179-186
12. Chawla, N.V., Bowyer,K.W., Hall, L.O., Kegelmeyer W.P.: SMOTE: Synthetic Minority Over-Sampling Technique. Journal of Artificial Intelligence Research 16 (2002) 321-357
13. Chawla, N.V., Lazarevic, A., Hall, L.O. and Bowyer, K.: SMOTEBoost: Improving prediction of the Minority Class in Boosting. 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat Dubrovnik, Croatia (2003) 107-119
14. Gustavo, E.A., Batista, P.A., Ronaldo, C., Prati, Maria Carolina Monard: A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. SIGKDD Explorations 6 (1) (2004) 20-29

15. Andrew Estabrooks, Taeho Jo and Nathalie Japkowicz: A Multiple Resampling Method for Learning from Imbalanced Data Sets. Comprtational Intelligence 20 (1) (2004) 18-36
16. Taeho Jo, Nathalie Japkowicz: Class Imbalances versus Small Disjuncts. Sigkdd Explorations 6 (1) (2004) 40-49
17. Hongyu Guo, Herna L Viktor: Learning from Imbalanced Data Sets with Boosting and Data Generation: The DataBoost-IM Approach. Sigkdd Explorations 6 (1) (2004) 30-39
18. Yoav Freund, Robert Schapire: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55 (1) (1997) 119-139
19. Joshi, M., Kumar,V., Agarwal, R.: Evaluating Boosting Algorithms to Classify Rare Classes: Comparison and Improvements. First IEEE International Conference on Data Mining, San Jose, CA (2001)
20. Gang Wu, Edward Y.Chang. Class-Boundary Alignment for Imbalanced Dataset Learning. Workshop on Learning from Imbalanced Datasets II, ICML, Washington DC (2003)
21. Kaizhu Huang, Haiqin Yang, Irwin King, Michael R. Lyu. Learning Classifiers from Imbalanced Data Based on Biased Minimax Probability Machine. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2004)
22. Dietterich, T., Margineantu, D., Provost, F. and P. Turney, edited, Proceedings of the ICML'2000 Workshop on Cost-sensitive Learning (2000)
23. Manevitz, L.M. and Yousef, M.: One-class SVMs for document classification. Journal of Machine Learning Research 2 (2001) 139-154
24. Blake, C., & Merz, C. (1998). UCI Repository of Machine Learning Databases http://www.ics.uci.edu/ ~ mlearn/ ~ MLRepository.html. Department of Information and Computer Sciences, University of California, Irvine
25. Quinlan, J. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA (1992)

.

Appendix C : Safe-Level-SMOTE

Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem

Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap

Department of Mathematics, Faculty of Science, Chulalongkorn University
chumphol@chiangmai.ac.th, Krung.S@chula.ac.th,
lchidcha@chula.ac.th

Abstract. The class imbalanced problem occurs in various disciplines when one of target classes has a tiny number of instances comparing to other classes. A typical classifier normally ignores or neglects to detect a minority class due to the small number of class instances. SMOTE is one of over-sampling techniques that remedies this situation. It generates minority instances within the overlapping regions. However, SMOTE randomly synthesizes the minority instances along a line joining a minority instance and its selected nearest neighbours, ignoring nearby majority instances. Our technique called Safe-Level-SMOTE carefully samples minority instances along the same line with different weight degree, called safe level. The safe level computes by using nearest neighbour minority instances. By synthesizing the minority instances more around larger safe level, we achieve a better accuracy performance than SMOTE and Borderline-SMOTE.

Keywords: Class Imbalanced Problem, Over-sampling, SMOTE, Safe Level.

1 Introduction

A dataset is considered to be imbalanced if one of target classes has a tiny number of instances comparing to other classes. In this paper, we consider only two-class case [5], [17]. The title of a smaller class is a minority class, and that of a bigger class is a majority class. The minority class includes a few positive instances, and the majority class includes a lot of negative instances.

In many real-world domains, analysts encounter many class imbalanced problems, such as the detection of unknown and known network intrusions [8], and the detection of oil spills in satellite radar images [13]. In these domains, standard classifiers need to accurately predict a minority class, which is important and rare, but the usual classifiers seldom predict this minority class.

Strategies for dealing with the class imbalanced problem can be grouped into two categories. One is to re-sample an original dataset [11], [14], [15], either by over-sampling a minority class or under-sampling a majority class until two classes are nearly balanced. The second is to use cost sensitive learning by assigning distinct costs to correctly classified instances or classifications errors [7], [9], [16].

T. Theeramunkong et al. (Eds.): PAKDD 2009, LNAI 5476, pp. 475–482, 2009.

Table 1. A confusion matrix for a two-class imbalanced problem

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

The performance of classifiers is customarily evaluated by a confusion matrix as illustrated in Table 1. The rows of the table are the actual class label of an instance, and the columns of the table are the predicted class label of an instance. Typically, the class label of a minority class set as positive, and that of a majority class set as negative. *TP*, *FN*, *FP*, and *TN* are True Positive, False Negative, False Positive, and True Negative, respectively. From Table 1, the six performance measures on classification; *accuracy*, *precision*, *recall*, *F-value*, *TP rate*, and *FP rate*, are defined by formulae in (1)-(6).

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FN} + \text{FP} + \text{TN}) . \tag{1}$$

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN}) . \tag{2}$$

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}) . \tag{3}$$

$$\text{F-value} = ((1 + \beta)^2 \cdot \text{Recall} \cdot \text{Precision}) / (\beta^2 \cdot \text{Recall} + \text{Precision}) . \tag{4}$$

$$\text{TP Rate} = \text{TP} / (\text{TP} + \text{FN}) . \tag{5}$$

$$\text{FP Rate} = \text{FP} / (\text{TN} + \text{FP}) . \tag{6}$$

The objective of a classifier needs to aim for high prediction performance on a minority class. Considering the definition of *accuracy*, if most instances in a minority class are misclassified and most instances in a majority class are correctly classified by a classifier, the *accuracy* is still high because the large number of negative instances influences the whole classification result on *accuracy*. Note that *precision* and *recall* are effective for this problem because they evaluate the classification rates by concentrating in a minority class. In addition, *F-value* [3] integrating *recall* and *precision*, is used instead of *recall* and *precision*. Its value is large when both *recall* and *precision* are large. The β parameter corresponding to relative importance of *precision* and *recall* is usually set to 1. Furthermore, ROC curve, The Receiver Operating Characteristic curve, is a standard technique for summarizing classifier performance over a range of tradeoffs between *TP rate*, benefits, and *FP rate*, costs. Moreover, AUC [2], Area under ROC, can also be applied to evaluate the performance of a classifier.

The content of this paper is organized as follows. Section 2 briefly describes related works for handling the class imbalanced problem. Section 3 describes the details of our over-sampling technique, Safe-Level-SMOTE. Section 4 shows the experimental results by comparing Safe-Level-SMOTE to SMOTE and Borderline-SMOTE. Section 5 summarizes the paper and points out our future works.

2 Related Works

Re-sampling is a preprocessing technique which adjusting the distribution of an imbalanced dataset until it is nearly balanced, before feeding it into any classifiers. The simplest re-sampling techniques are a random over-sampling technique [14] and a random under-sampling technique [14]. The former randomly duplicates positive instances into a minority class, while the latter randomly removes negative instances from a majority class. Both techniques are sampling the dataset until the classes are approximately equally represented. However, the random over-sampling technique may cause the overfitting problem [19] because the technique may create the decision regions smaller and more specific. The random under-sampling technique encounters the problem that diminishing some important information of a dataset. For handling these problems, improved re-sampling techniques were studied and are described as follows.

Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W. (2002) designed the State of the Art over-sampling technique, namely SMOTE, Synthetic Minority Over-sampling TEchnique [4]. It over-samples a minority class by taking each positive instance and generating synthetic instances along a line segments joining their k nearest neighbours in the minority class. This causes the selection of a random instance along the line segment between two specific features. 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 now learned for positive instances, rather than those being subsumed by negative instances around them. The effect is that decision trees generalize better. However, SMOTE encounters the overgeneralization problem. It blindly generalizes the region of a minority class without considering a majority class. This strategy is particularly problematic in the case of highly skewed class distributions since, in such cases, a minority class is very sparse with respect to a majority class, thus resulting in a greater chance of class mixture.

Han, H., Wang, W., Mao, B. (2005) designed the improvement of SMOTE, namely Borderline-SMOTE [10]. The authors divided positive instances into three regions; noise, borderline, and safe, by considering the number of negative instances on k nearest neighbours. Let n be the number of negative instances among the k nearest neighbours. The three regions are defined by the definitions in Table 2. Borderline-SMOTE uses the same over-sampling technique as SMOTE but it over-samples only the borderline instances of a minority class instead of over-sampling all instances of the class like SMOTE does. Unfortunately, considering two positive instances those n values are equal to k and k-1 for the first and second instances consecutively. These instances are not obviously difference but they are divided into the different regions; noise and borderline. The first instance is declined but the second instance is selected for over-sampling.

Table 2. The definitions of noise, borderline, and safe regions in Borderline-SMOTE

Region	Definition
Noise	$n = k$
Borderline	$\frac{1}{2}k \leq n < k$
Safe	$0 \leq n < \frac{1}{2}k$

3 Safe-Level-SMOTE

Based on SMOTE, Safe-Level-SMOTE, Safe-Level-Synthetic Minority Over-sampling TEchnique, assigns each positive instance its *safe level* before generating synthetic instances. Each synthetic instance is positioned closer to the largest *safe level* so all synthetic instances are generated only in safe regions.

The *safe level* (*sl*) is defined as formula (7). If the *safe level* of an instance is close to 0, the instance is nearly noise. If it is close to *k*, the instance is considered safe. The *safe level ratio* is defined as formula (8). It is used for selecting the safe positions to generate synthetic instances.

$$\text{safe level } (sl) = \text{the number of a positive stances in } k \text{ nearest neighbours .} \quad (7)$$

$$\text{safe level ratio} = sl \text{ of a positive instance} / sl \text{ of a nearest neighbours .} \quad (8)$$

Safe-Level-SMOTE algorithm is showed in Fig. 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. *p*, *n*, and *s* are vectors. sl_p, sl_n, *sl_ratio*, *numattrs*, *dif*, and *gap* are scalars.

After assigning the *safe level* to *p* and the *safe level* to *n*, the algorithm calculates the *safe level ratio*. There are five cases corresponding to the value of *safe level ratio* showed in the lines 12 to 28 of Fig. 1.

The first case showed in the lines 12 to 14 of Fig. 1. 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 noises. 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 showed in the lines 17 to 19 of Fig. 1. The *safe level ratio* is equal to ∞ and the *safe level* of *p* is not equal to 0. It means that *n* is noise. 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 showed in the lines 20 to 22 of Fig. 1. 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*.

The fourth case showed in the lines 23 to 25 of Fig. 1. 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* because *p* is safer than *n*. The synthetic instance will be generated in the range [0, 1 / *safe level ratio*].

The fifth case showed in the lines 26 to 28 of Fig. 1. 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* because *n* is safer than *p*. The synthetic instance will be generated in the range [1 - *safe level ratio*, 1].

After each iteration of *for* loop in line 2 finishes, if the first case does not occurs, a synthetic instance *s* will be generated along the specific-ranged line 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 instances that satisfy the first case.

```
Algorithm: Safe-Level-SMOTE
Input: a set of all original positive instances D
Output: a set of all synthetic positive instances D'
 1. D' = ∅
 2. for each positive instance p in D {
 3.    compute k nearest neighbours for p in D and
       randomly select one from the k nearest neighbours, call it n
 4.    sl_p = the number of positive stances in k nearest neighbours for p in D
 5.    sl_n = the number of positive stances in k nearest neighbours for n in D
 6.    if (sl_n ≠ 0) {  ; sl is safe level.
 7.      sl_ratio = sl_p / sl_n  ; sl_ratio is safe level ratio.
 8.    }
 9.    else {
10.      sl_ratio = ∞
11.    }
12.    if (sl_ratio = ∞ AND sl_p = 0) {  ; the 1st case
13.      does not generate positive synthetic instance
14.    }
15.    else {
16.      for (atti = 1 to numattrs) {  ; numattrs is the number of attributes.
17.        if (sl_ratio = ∞ AND sl_p ≠ 0) {  ; the 2nd case
18.          gap = 0
19.        }
20.        else if (sl_ratio = 1) {  ; the 3rd case
21.          random a number between 0 and 1, call it gap
22.        }
23.        else if (sl_ratio > 1) {  ; the 4th case
24.          random a number between 0 and 1/sl_ratio, call it gap
25.        }
26.        else if (sl_ratio < 1) {  ; the 5th case
27.          random a number between 1-sl_ratio and 1, call it gap
28.        }
29.        dif = n[atti] - p[atti]
30.        s[atti] = p[atti] + gap·dif
31.      }
32.      D' = D' ∪ {s}
33.    }
34. }
35. return D'
```

Fig. 1. Safe-Level-SMOTE algorithm

4 Experiments

In our experiments, we use four performance measures; *precision*, *recall*, *F-value*, and AUC, for evaluating the performance of three over-sampling techniques; Safe-Level-SMOTE, SMOTE, and Borderline-SMOTE. The value of β in *F-value* is set to 1 and the value of k in all over-sampling techniques are set to 5. The performance measures are evaluated through 10-fold cross-validation. Three classifiers; decision trees C4.5 [18], Naïve Bayes [12], and support vector machines (SVMs) [6], are applied as classifiers in the experiments. We use two quantitative datasets from UCI Repository of Machine Learning Databases [1]; Satimage and Haberman, illustrated in Table 3. The first to last column of the table represents the dataset name, the number of instances, the number of attributes, the number of positive instances, the number of negative instances, and the percent of a minority class, respectively.

The experimental results on the two datasets are illustrated in Fig. 2. The x-axis in these figures represents the over-sampling percent on a minority class. The y-axis in these figures represents the four performance measures; *precision*, *recall*, *F-value*, and AUC, in order from Fig. 2 (a) to Fig. 2 (d). In these figures, ORG, SMOTE, BORD, and SAFE are the label of the original dataset, SMOTE, Borderline-SMOTE, and Safe-Level-SMOTE, respectively.

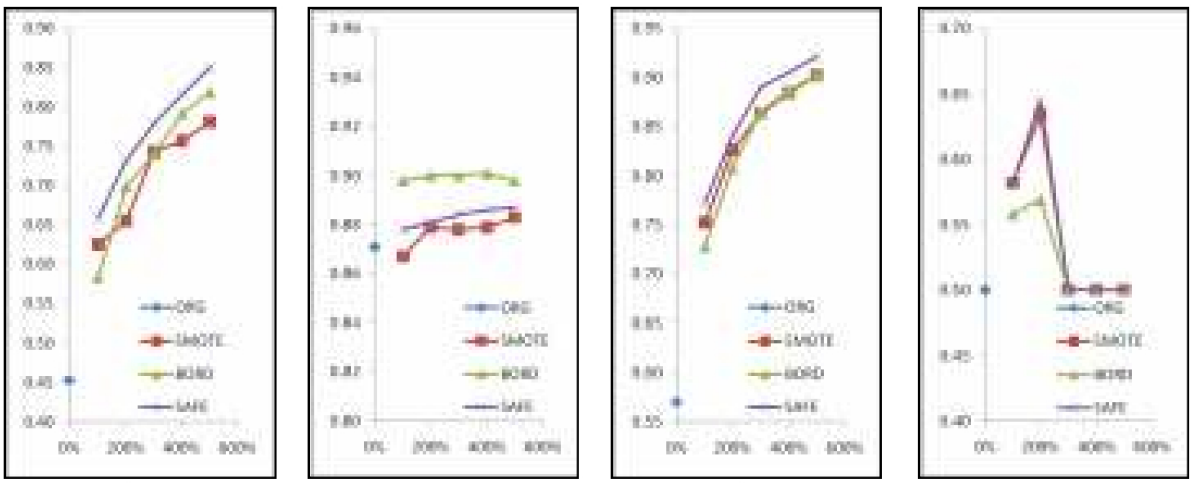
For Satimage dataset, we select the class label 4 as the minority class and merge the remainder classes as the majority class because we only study the two-class imbalanced problem. The results on *F-value* using decision trees C4.5 are illustrated in Fig. 2 (c). It is apparent that *F-value* is improved when over-sampling percent on the minority class is increased. Moreover, Safe-Level-SMOTE achieved higher *F-value* than SMOTE and Borderline-SMOTE. The results on *recall* using Naïve Bayes are illustrated in Fig. 2 (b). Analyzing the figure, Borderline-SMOTE gains the higher performance on *recall*, while Safe-Level-SMOTE comes second.

For Haberman dataset, the minority class is about one quarter of the whole dataset. The results on *precision* using decision trees C4.5 are illustrated in Fig. 2 (a). The performance of Safe-Level-SMOTE is the best performance on *precision*. The results on AUC using SVMs are illustrated in Fig. 2 (d). Analyzing the figure, Safe-Level-SMOTE and SMOTE show similar performance on AUC. In addition, Borderline-SMOTE shows poor performance on higher percent.

For all experimental results, Safe-Level-SMOTE obviously achieve higher performance on *precision* and *F-value* than SMOTE and Borderline-SMOTE when decision trees C4.5 are applied as classifiers. Borderline-SMOTE only achieve a better performance on *recall* when Naive Bayes are applied as classifiers since the independent assumption on the borderline region is valid. Moreover, the SVMs show no improvement in all over-sampling techniques. Theses causes by the convex regions of all over-sampling techniques are similar. Therefore, the results of hyperplanes in SVMs are indistinguishable.

Table 3. The descriptions of UCI datasets in the experiments

Name	Instance	Attribute	Positive	Negative	% Minority
Satimage	6,435	37	626	5,809	9.73
Haberman	306	4	81	225	26.47



(a) (b) (c) (d)

Fig. 2. The experimental results; (a) Precision evaluated by applying C4.5 with Haberman, (b) Recall evaluated by applying Naïve Bayes with Satimage, (c) F-value evaluated by applying C4.5 with Satimage, (d) AUC evaluated by applying SVMs with Haberman

5 Conclusion

The class imbalanced problem has got more attentions among data miners. There are many techniques for handling such problem. However, traditional data mining techniques are still unsatisfactory. We present an efficient technique called Safe-Level-SMOTE to handle this class imbalanced problem.

The experiments show that the performance of Safe-Level-SMOTE evaluated by *precision* and *F-value* are better than that of SMOTE and Borderline-SMOTE when decision trees C4.5 are applied as classifiers. This comes from the fact that Safe-Level-SMOTE carefully over-samples a dataset. Each synthetic instance is generated in safe position by considering the *safe level ratio* of instances. In contrast, SMOTE and Borderline-SMOTE may generate synthetic instances in unsuitable locations, such as overlapping regions and noise regions. We can conclude that synthetic instances generated in safe positions can improve prediction performance of classifiers on the minority class.

Although the experimental results have provided evidence that Safe-Level-SMOTE can be successful classified numeric datasets in the class imbalanced problem, there are several future works left to be studied in this line of research. First, different definitions to assign *safe level* would be valuable. Second, additional methods to classify datasets which have nominal attributes are useful. Third, automatic determination of the amount of synthetic instances generated by Safe-Level-SMOTE should be addressed.

References

1. Blake, C., Merz, C.: UCI Repository of Machine Learning Databases. Department of Information and Computer Sciences, University of California, Irvine, CA, USA (1998), `http://archive.ics.uci.edu/ml/`
2. Bradley, A.: The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms. Pattern Recognition 30(6), 1145–1159 (1997)

3. Buckland, M., Gey, F.: The Relationship between Recall and Precision. Journal of the American Society for Information Science 45(1), 12–19 (1994)
4. Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W.: SMOTE: Synthetic Minority Over-Sampling Technique. Journal of Artificial Intelligence Research 16, 321–357 (2002)
5. Chawla, N., Japkowicz, N., Kolcz, A.: Editorial: Special Issue on Learning from Imbalanced Data Sets. SIGKDD Explorations 6(1), 1–6 (2004)
6. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, Cambridge (2000)
7. Domingos, P.: Metacost: A General Method for Making Classifiers Cost-sensitive. In: The 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 1999), pp. 155–164. ACM Press, San Diego (1999)
8. Fan, W., Miller, M., Stolfo, S., Lee, W., Chan, P.: Using Artificial Anomalies to Detect Unknown and Known Network Intrusions. In: The 1st IEEE International Conference on Data Mining (ICDM 2001), San Jose, CA, USA, pp. 123–130 (2001)
9. Fan, W., Salvatore, S., Zhang, J., Chan, P.: AdaCost: misclassification cost-sensitive boosting. In: The 16th International Conference on Machine Learning (ICML 1999), Bled, Slovenia, pp. 97–105 (1999)
10. Han, H., Wang, W., Mao, B.: Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In: Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) ICIC 2005. LNCS, vol. 3644, pp. 878–887. Springer, Heidelberg (2005)
11. Japkowicz, N.: The Class Imbalance Problem: Significance and Strategies. In: the 2000 International Conference on Artificial Intelligence (IC-AI 2000), Las Vegas, NV, USA, pp. 111–117 (2000)
12. Kamber, M., Han, J.: Data mining: Concepts and Techniques, 2nd edn., pp. 279–327. Morgan-Kaufman, NY, USA (2000)
13. Kubat, M., Holte, R., Matwin, S.: Machine Learning for the Detection of Oil Spills in Satellite Radar Images. Machine Learning 30, 195–215 (1998)
14. Kubat, M., Matwin, S.: Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In: The 14th International Conference on Machine Learning (ICML 1997), pp. 179–186. Morgan Kaufmann, Nashville (1997)
15. Lewis, D., Catlett, J.: Uncertainty Sampling for Supervised Learning. In: The 11th International Conference on Machine Learning (ICML 1994), pp. 148–156. Morgan Kaufmann, New Brunswick (1994)
16. Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., Brunk, C.: Reducing Misclassification Costs. In: The 11th International Conference on Machine Learning (ICML 1994), pp. 217–225. Morgan Kaufmann, San Francisco (1994)
17. Prati, R., Batista, G., Monard, M.: Class Imbalances versus Class Overlapping: an Analysis of a Learning System Behavior. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) MICAI 2004. LNCS (LNAI), vol. 2972, pp. 312–321. Springer, Heidelberg (2004)
18. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1992)
19. Tetko, I., Livingstone, D., Luik, A.: Neural network studies. 1. Comparison of Overfitting and Overtraining. Chemical Information and Computer Sciences 35, 826–833 (1995)

Biography

Name Surname	Jatuporn Rueangkhetpit
Date of Birth	22 August 1995
Address	House No. 226/3 Village No. 4
	Khao Bang Kraek sub-district,
	Nong Chang district, Uthai Thani province 61170

Education Background

Year 2014	Senior High School
	Nongchang Wittaya School, Uthai Thani province
Year 2018	Bachelor of Science (Mathematics)
	University of Phayao, Phayao province
Telephone number	086-2178027
E-mail	nanjatuporn16520@gmail.com

Name Surname	Saranya Pagorntadapan
Date of Birth	22 November 1995
Address	House No. 36 Village No. 3
	Nang Lae sub-district, Muang Chiang Rai district,
	Chiang Rai province 57100

Education Background

Year 2014	Senior High School
	Damrongratsongkroh School, Chiang Rai province
Year 2018	Bachelor of Science (Mathematics)
	University of Phayao, Phayao province
Telephone number	089-4320510
E-mail	pagorn.pear@gmail.com

Name Surname	Inprapa Aunsanee
Date of Birth	12 November 1995
Address	House No. 176 Village No. 6
	Rim Kok sub-district, Muang Chiang Rai district,
	Chiang Rai province 57100

Education Background

Year 2014	Senior High School
	Damrongratsongkroh School, Chiang Rai province
Year 2018	Bachelor of Science (Mathematics)
	University of Phayao, Phayao province

Telephone number	088-8057018
E-mail	oilly.inprapa@gmail.com