Hybrid classification approach of SMOTE and instance selection for imbalanced datasets

by

Tianxiang Gao

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Industrial Engineering

Program of Study Committee:

Sigurdur Olafsson, Major Professor

Sarah Ryan

Dianne Cook

Iowa State University

Ames, Iowa

2015

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The research area of imbalanced dataset has been attracted increasing attention from both academic and industrial areas, because it poses a serious issues for so many supervised learning problems. Since the number of majority class dominates the number of minority class are from minority class, if training dataset includes all data in order to fit a classic classifier, the classifier tends to classify all data to majority class by ignoring minority data as noise. Thus, it is very significant to select appropriate training dataset in the prepossessing stage for classification of imbalanced dataset. We propose an combination approach of SMOTE (Synthetic Minority Over-sampling Technique) and instance selection approaches. The numeric results show that the proposed combination approach can help classifiers to achieve better performance.

CHAPTER 1. INTRODUCTION

Recently, with the development of technology and science, data growths explosively, which has proposed an enormous opportunity for data mining research since it has a broad range of application from daily life to governmental decision-making system. In Knowledge discovery and data mining area, problems are naturally divided into two categories, supervised learning and unsupervised learning. The essential difference between those two is that supervised learning has a supervised respond variable, whereas unsupervised learning does not have any respond variable rather only several input variables. Moreover, supervised learning are further separated into two sub-classical problems, classification and regression problems, based on the properties of respond variables. If respond variables are discrete or categorical, these are classification problems. Otherwise, if respond variables are continuous variables, they are regression problems.

Typically, a classification problem is to classify previously unseen observations, which is know as testing dataset by utilization of a classifier that is trained by previously given observations, called training data set. Several standard classifiers have been proposed to deal with this type of problems such as logistic regression, discriminant analysis, and tree-based classifiers. Simply, take decision tree as an example to clearly introduce the process of classification. Before introducing decision tree in detail, another essential concept needed to be introduced is the error rate. It is widely utilized to evaluate the performance of a classifier. Regular accuracy is a well-known error rate used in a wide area, that is, the sum of total misclassified observations is divided by the total number of instances. Decision tree separates data space in an top-down fashion by iteratively selecting the best attribute (or variable) so as to reduce the error rate. Specifically, starting with the root node, in each iteration, a decision tree selects an appropriate value of an associated attribute to split the whole data pace. As result, the error rate reduces largest in this iteration and that value of attribute is considered as a node. As the tree grows up, the test error keeps reducing. If there is no restriction to the growth of a decision tree, the decision tree would output a huge tree with a single data point in each note. That would cause overfitting because the decision tree is constructed in training data set, while in realistic application, we generally expect a good performance of a classifier in previously unseen data set, which is test data set.

The overfitting scenario appears in various classifiers, specially for given data set is imbalanced data set. Simply, we consider binary classification problems. In imbalanced datasets, number of minority class is dominated by number of majority class. As result, a classifier classifies all data into majority class and almost ignores minority class. A lot of approaches have been proposed to address this scenario. Sampling is a well-known prepossessing strategy to improve the performance of classifiers. It tries to balance the distribution of classes by oversampling or undersampling the number of minority class or majority class. On another hand, modification of current standard classifier is also able to conquer this type of problems such as support vector machine and boosting. We propose another prepossessing method that is combination of oversampling and instance selection technique. Instance selection has been widely implemented in machine learning area. It selects subset of training dataset in order to remove superfluous instances and contain satisfactory accuracy. The selected subset allows classifiers to have reduced time to learn and even have better performance.

The following organization of this thesis is divided into 4 chapters. A literature review associated with nature of imbalanced datasets, well-known sampling methods, and popular instance selection techniques will be covered in Chapter 2. Chapter 3 will introduce the formula and pseudo-code of our combination method of SMOTE and Instance selection in detail. In Chapter 4, several numeric results and comparison with other sampling and instance selection methods will be presented. Finally, in Chapter 5, the conclusion and future research will be given.

CHAPTER 2. REVIEW OF LITERATURE

2.1 Issues of Imbalanced Data Sets

Generally speaking, any dataset can be considered as imbalanced dataset if the number of observations between classes are not equal. However, the common understanding of imbalanced dataset is that a dataset exhibits significant, and even extreme imbalanced. The *imbalanced ratio* is about at least 1:10. Even though there are several cases of multiclass datasets, we in this thesis consider binary (or two-class) cases.

Preferably, given any dataset, we typically require a standard classifier to provide balanced weighs of predictive accuracy for both minority and majority class. In practice, the standard classifier tends to provide an extreme imbalanced prediction accuracy, minority class has accuracy usually less than 10 percent, while majority class has accuracy close to 100 percent(Chawla et al., 2002). However, in application of imbalanced data set (Chawla et al., 2002), we expect a classifier is sensitive to minority class as those are the class we are interested, even though pay certain penalty to missclassify the majority class. Therefore, we prefer a classifier that identifies most of minority observations, while remains relative high accuracy of majority class. Further, regular accuracy as conventional assessment metrics does not provide enough helpful information when dealing with imbalanced data set. More informational assessment metrics are needed like Receiver Operating Characteristics curve.

The common understanding of imbalanced dataset in community is referred to as *intrinsic* imbalanced dataset, that is, imbalanced dataset is as result from the nature of the data space.The domain of imbalanced dataset, however, is not restricted by just intrinsic variety. Dataset can be considered as *extrinsic* imbalanced (He and Garcia, 2009) if it has time or storage variables. By extrinsic imbalanced, the imbalanced is not directly as a result of nature of data space. For example, a continuous balanced data stream has been kept the training dataset balanced for a while until during a time interval, the transmission has temperately interrupted, then the acquired dataset can be possibly imbalanced. Figure 2.1 is an such example. During time internal of 4.85 and 7.99, class 2 becomes minority class as the its data stream has been temperately interrupted.

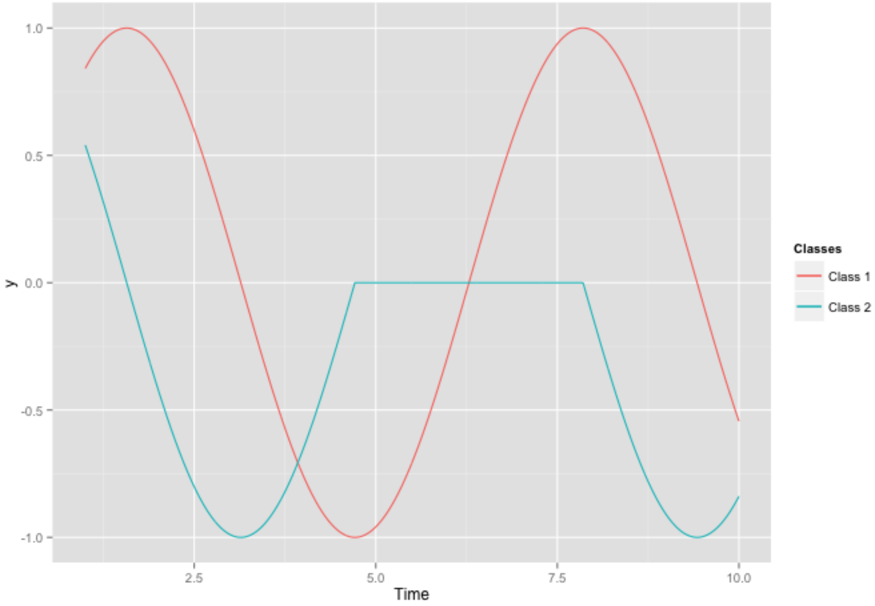

Figure 2.1: An example of extrinsic imbalance

In addition, *relative imbalance* and *absolute imbalance* are totally different concepts. Absolute imbalance is due to rare observations, while relative imbalance is imbalance relative to other class. For instance, a given dataset with imbalance ratio $1 : 100$ contains $200,000$ observations. It is no doubt that the number of majority class dominates the number of minority class. However, $2,000$ observations of minority class is not really rare. instead the number of minority class is rare relative to the majority class. A number of papers have shown that the minority class is not so hard to be identified for relative imbalanced dataset. (Batista et al., 2004). These results are very suggestive. Hence, imbalance ratio is just one of factors hindering classifiers. As a result, *dataset complexity* plays a critical role in classification deterioration.

Dataset complexity is referred to *overlapping*, and *small disjucts*. It is not uncommon that a dataset with more than one classes has overlapping, that is, various class data overlap

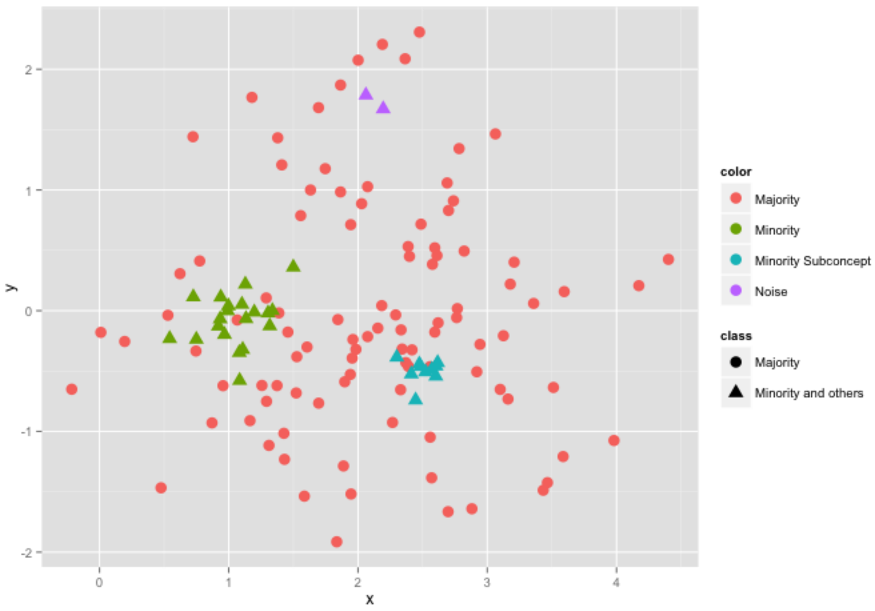

Figure 2.2: An example of within-class imbalance

in the neighborhood of decision boundary. In Figure 2.2, there are data points overlapped between majority class which is circle and minority class which is treated as triangle. Moreover, the minority class additionally contain a set of *subconcept* data points which are colored as pure blue. This is referred to another situation of imbalance, *within-class imbalance* (Jo and Japkowicz, 2004). The existence of within-imbalance is closely related with small disjuncts, which greatly depreciate classification performance (Jo and Japkowicz, 2004). Interpretation to small disjuncts is that a classifier is required to detect the minority class (or majority class) by creating multiple disjunct rules that individually describes the specific subconcept of minority class. For instance, removing all data of subconcept of minority class in Figure 2.2, a classifier will generally and easily create a single large disjunct that cover a large piece of observations associated with the main concept. However, because of the existence of minority subconcept, the classifier instead is required to proved at lease two small disjuncts that individually covers a specific data space of minority class. Moreover, noise can influence disjuncts of minority class if considered as a small disjunct of minority class. Hence, the validity of data becomes an essential issue, that is, whether a small disjunct corresponds to an actual subconcept of class or are merely noise. In Figure 2.2, the two purple noise are considered as minority class by miss classified.

Combination of imbalanced data and small sample size is the a comprehensive issue associated with imbalanced dataset. In realistic such as face recognizance, it is inevitable that the number of predictors are far more than number of instances, i.e., $p >> n$. Thus, the first issue related is, since the number of samples is pretty limited, all of issues associated with absolute imbalance and within-imbalance is applicable. Second, it is harder for classifier to find a inductive role to cover the minority class. Furthermore, the classifier goes to overfiting if generates specific rule because of limited samples of minority class in specific data space.

2.2 Sampling Methods for Imbalanced Datasets

Technically speaking, sampling methods consists of modification on distribution of a imbalanced dataset. A number of studies have shown that a balanced dataset improves performance of some base classifiers compared with imbalanced dataset (e.g. Weiss and Provost (2001), Estabrooks et al. (2004)).

Oversampling with replacement modifies on distribution of data by appending observations to minority class, that is, randomly selects minority observations in its original dataset with replacement(Ling and Li, 1998). While oversampling increases the number of minority observations, undersampling removes observations from the training dataset. Concretely, randomly remove a subset of majority observations from the original training dataset. Undersampling is simple and has additional benefit such as reducing time consuming.

At first glance, those two sampling methods seems to be pretty much the same since they both consist of modification on the distribution of data and focus on class distribution of data space. However, this commonality is peripheral since individual method has its own negative effect on hindering learning. In undersampling case, it is obvious that randomly removing subset of majority class from the original data set probably results in the classifier to miss important observations of majority class. This issue is relatively easy to solve if systematically removing those majority class data instead. On another word, keep or select those important instances from majority class that determining the decision boundary between the binary class. Instance selection technique is capable to cover this issue by selecting those importance instances and this technique will be deeply discussed in the latter section of this chapter. On another hand, oversampling with replicated instances literately not very comparable with undersampling method because duplicated minority data make the decision boundary so specific that cause base classifiers easily to go overfitting, even the training accuracy is very high.

A representative synthetic sampling method is synthetic oversampling minority technique, which is so powerful technique that has been shown a great successful in varied studies of imbalance datasets(Chawla et al., 2002). SMOTE generates synthetic samples of minority class based on feature space instead of data space in randomly sampling methods. Concretely, take each instance of minority class, x_i, in original data set and compute its K nearest neighbors, which is defined as the smallest Euclidian distance between itself and other minority data. To generate a synthetic sample of minority class, randomly pick up one of the nearest neighbors, $\widehat{x_i}$, and calculate the p-dimensional feature difference vector $\widehat{x_i} - x_i$, assume the there are p predictors in dataset. The new synthetic sample, x_new, is achieved by multiplying a random numberδ between $[0, 1]$, and finally adding x_i:

$$x_{new} = x_i + (\widehat{x_i} - x_i) \times \delta \tag{2.1}$$

Therefore, the new synthetic data, x_{new}, is an instance that is along the line segment between x_i and x_{new}. This procedure has a potential benefit that does not cause necessarily validity of the class which discussed in previous section. As the created synthetic data lies in the line segment jointing x_i and randomly selected nearest neighbor $\widehat{x_i}$, it is more likely to be the minority class instead of noise. SMOTE also has its drawbacks, such as overgeneralization since generation of synthetic samples increases the occurrence of overlapping between classes (He et al., 2008). Overgeneralization can be easily solved by instance selection since this method only selects subset and best representative instances of data set.

2.3 Instance Selection

Classification problem is to classify the previously unseen testing dataset based on a classifier that is trained by a given training dataset T. In practice, however, a training dataset contains useless information such surplus observations that are noisy or redundant. Hence, a process is required that discards those surplus observations from a training dataset and groups the left important observations as the final training data set S.

Typically, instance selection is to select subset of a training dataset such that the subset S removes those surplus observations, in the mean while maintains the accuracy as the whole training dataset. Instance selection methods basically start with either empty space (incremental method) or whole space (decremented method) with entire data point from T. The difference between those two is incremental method adds instances to T during the selection process, whereas decremented method removes instances from S along selection.

Instance selection are similar to feature selection in term of strategies. We are able to divide instance selection methods into two main categorical, wrapper and filter (Olvera-López et al., 2010b). Wrapper methods select those subset instances based on the performance of a predefined classifier, that is, those instances are discarded from T if they do not contribute appropriate information for that classifier. Filter methods instead rank the instances in T based on a non-classifier based function, then select the target subset from the ranking. Comparing with wrapper methods, filter are faster, in the mean while deliver competitive accuracy and retention, especially for processing medium or large data sets. However, wrapper methods have broader utility than filter methods because of their classifier-based customizability.

2.3.1 Wrapper Methods

K nearest neighborhood (Cover and Hart, 1967) has been broadly used for various wrapper methods. The earliest wrapper method is known as *Condensed Nearest Neighbor (CNN).* It initially randomly selects a single instance belonging to each class in T and trains a classifier using those two instances. After initialization, S combines those misclassified instances by the classifier. As the number of instances are increasing, this kind of methods are known as *incremental method.* The *Generalized Condensed Nearest Enighbor rule (GCNN)* (Chou et al., 2006) is an extension of CNN is . Basically speaking, $GCNN$ and CNN are functionally equal. However, $GCNN$ introduces a concept of *absorption* and selects the target subset of instances S based absorption and a predefined threshold. In selection process, the instance $p \in T$ would not be included in S if it is absorbed (or represented) by S, while its absorption criteria is computed according to its nearest neighbor and nearest enemy (an instance from different class). Specifically speaking, if $|p - x| - |p - y| > \delta$, then the instance p would be absorbed, where $xy \in S$ are the nearest neighbor and enemy.

Another series of wrapper methods involved K *nearest neighborhood* are a set of methods proposed by Wilson and Martinez (2000). *DROP 1-5 (Decremental Reduction Optimization Procedure)*are *decremental methods* that introduce a concept of *associate*. For each instance p, define its associates as a set of instances such that p belongs to their K *nearest neighborhood.* Specifically, $DROP1$ starts with $S = T$ and removes any instance p from S if associates of p can be correctly classified without p. Hence, noisy instances are more likely to be discarded since their associates are commonly classified correctly. On the other hand, if associates of a noisy instance have been removed before its removing, the noisy instance will retain in S. In order to conquer this issue, $DROP2$ extends the broaden the associates of p to whole S, that is the instance p will be removed if all instances in S are correctly classified. $DROP3$ and $DROP4$ similarly removes noisy instances firstly. If a instance is misclassified by its K *nearest neighborhood*, this instance is considered as a noise and removed. After that, apply $DROP2$. $DROP5$ is based on $DROP2$ but it removes nearest enemies so as to have smoother decision boundaries.

In 2002, tabu search (Glover, 1986) was applied by Zhang and Sun for selecting the optimal or close-optimal instances (solution). Tabu search starts with an initial set of instances (an initial feasible solution) $S_i \in T$. In selection process, the nearest neighbor subset S_j of S_i that is subset differs from S_i just one instance, is evaluated by a classifier trained from S_i. S_j will replace S_i if it has better accuracy or performance for the classifier, otherwise, declared tabu. After all iterations, the optimal solution is subset S. One characteristic of tabu search is that in selection process any classifier can be used for selecting S.

Evolutionary algorithm (Kuncheva 1995, and 1997,Bezdek and Kuncheva 2001, and Cano et al. 2003) has the same characteristic as tabu search. The evolutionary algorithm generates the the initial population of individuals (an initial set of instances). Initials are evaluated by the fitness function (a classifier). Select the best chromosomes that maximize the fitness function (in instance selection context, improve the performance of classifier) and undergo mutation, selection, and recombination. Typically, a chromosome is represented as a binary n-dimensional variable, $C = [0, 1, ..., 1]^T$. C_i means the $i$the element in training data set T, and is selected into the subset if $C_i = 1$. According to the requirement of user, the algorithm is repeated several iterations (generations) and the last generation is selected.

2.3.2 Filter Methods

Unlike wrapper methods, filter methods determine an instance is whether discarded from the training data set T based on the performance of predefined classifier.

In training dataset, any instance can be considered as either interior instances or a border instance (or support instance). We assume that it is binary classification scenario, an instance $p_i \in t$ is belonging to a border instance if $p_i \in C_0$ is one of the $Knearestneighbors$ of an instance from class C_1. On another word, a border instance is an instance in the decision boundary neighborhood. Besides border instances, the left instances are interior instances. Hence, a number of filter methods have been proposed based on selection of border instances.

Raicharoen and Lursinsap (2005) proposed the POC-NN (Pair Opposite Class-Nearest Neighbor) method that discards interior instances and selects border instances. In selection process, POC-NN computes the mean of instances in each class, say m_1 and m_2. In order to select a border instance, computes the distances of each individual instance $p_i \in C_0$ between itself and the mean m_2 of another class. The instance p_b with the shortest distance is selected as it is the nearest neighbor to m_2. The border instances are also selected from class $C2$ based on the same selection process.

Another filter method *Object Selection by Clustering (OSC)* was proposed by Olvera-López et al. 2010a, which is not based on *K nearest neighborhood*, instead, utilizes Clustering instance selection (Lumini and Nanni, 2006). OSC initially divides T into K subsets by clustering.

Those subsets always have two categories, homogeneous and non-homogeneous. Several interior instances in homogeneous subsets will retain in S if it is the nearest neighbor of the mean point of that subset. In non-homogeneous subsets, those important border instances, p, will be combined into S if p is the nearest neighbor of an enemy instance which belongs to a different class.

CHAPTER 3. METHODS AND PROCEDURES

3.1 SMOTE - Synthetic Minority Oversampling Technique

The use of sampling methods in applications of imbalanced datasets is composed of the modifications of amounts of classes data in order to provide a balanced class distribution. Chawla et al. (2002) proposed a powerful over-sampling approach called SMOTE, which stands for Synthetic Minority Oversampling Technique. This approach has been widely cited and obtain great successful when dealing with the applications of imbalanced datasets.

Before Chawla et al. give birth to SMOTE, several researchers proposed various sampling approaches to conquer the problems of imbalanced datasets . Kubat et al. (1997) created their own training datasets by selectively under-sampling the number of data points of majority class while keeping the original population of the minority class. Under-sampling the majority class enables better classifiers to be built. Another approach that is really relevant to Chawla et al.'s work is from Ling and Li (1998). However, over-sampling the minority class with replacement did not give them better classifiers than the under-sampling approach. Moreover, a combination of previous two sampling approaches did not lead to classifiers that outperform those built utilizing only under-sampling. Hence,Over-sampling the minority class doesn't significantly improve the recognition of minority class in terms of decision regions in feature space. Essentially speaking, as the amount of minority class is increased by over-sampling with replacement, the effect for classifiers is to identify similar and even more specific regions in the feature space, even though the data space of imbalanced datasets has been modified.

SMOTE uses completely different method to over-sample the minority class, which operates on the feature space rather than data space. Basically, SMOTE was designed to deal with continuous (or discrete) variables. Given an imbalanced dataset, the minority class is over-sampled by taking each minority class instance and creating amount of synthetic samples along with the K minority class nearest neighbors. The parameter N which is amount of synthetic samples generated per original minority instance, and the parameter K related to nearest neighbors need to be predefined. Typically, people uses $N = 200\%$ amount of over-sampled samples and five nearest neighbors. For example, $N/100$ neighbors from the five nearest neighbors are chosen, and one synthetic sample is generated in the direction of each. Specifically, the procedure of generating synthetic samples consists of the following steps: S_{min} is a set of minority class from $\subseteq S$, for each instance $x_i \in S_{min}$, find its K nearest neighbors by using Euclidean distance. To create a synthetic sample, randomly select one of the K-nearest neighbors, calculate a feature difference between x_i and its neighbor. Furthermore, multiply this feature vector difference by a random number $\alpha \in [0, 1]$, and finally add this vector to x_i to get the synthetic sample x_{new}

$$x_{new} = x_i + (\widehat{x_i} - x_i) \times \alpha \tag{3.1}$$

where $x_i \in S_{min}$ is an instance of minority class in original population, $\widehat{x_i}$ is one of the K-nearest neighbors of x_i, and $\alpha \in [0, 1]$ is a real random number. Therefore, the new synthetic sample based on (3.1) is a data point along the line segment between x_i and the randomly selected K nearest neighbor $\widehat{x_i}$. The synthetic samples causes classifiers to create larger and less specific decision regions for minority class. The algorithm 1 is the pseudo code for SMOTE.

Typically, STOME is a powerful technique proposed to conquer imbalanced datasets with numerical variables originally. However, categorical (or discrete, nominal) variables arise in a variety of application of imbalanced datasets. As we randomly pick one neighbor x_{i_β} after computing K nearest neighbors for instance x_i, we could just allocate the nominal feature value of that neighbor x_{i_β} to the new synthetic sample. This is the easiest way to deal with categorical variables. On the other hand, majority vote is another simple and straight strategy for categorical variables, but implement more diversity into the algorithm. Hence, in the processing of creating new synthetic samples, take majority vote between feature vector under consideration and K nearest neighbors for the categorical feature values. In the case of a tie, random choose. Based on this strategy, a variant of SMOTE (Chawla et al., 2003) are proposed

Algorithm 1: SMOTE

```
Input: $T$:Training set; $N$ : $N\%$ amount of synthetic samples ; $k$: Number of nearest
       neighbors.
Output: $S$: Set of synthetic samples
begin
    $X \longleftarrow$ MinorityInstances(T);
    $n \longleftarrow$ NumberOfInstances(X);
    $p \longleftarrow$ NumberOfVariables(X);
    $S \longleftarrow \emptyset$;
    if $N < 100$ then
        $n \longleftarrow (N/100) \times n$ ;                         /* $N\%$ will be SMOTEd */
        $X \longleftarrow$ RandomSample($X, p$);
        $N \longleftarrow 100$;
    end
    $N \longleftarrow N/100$;
    for $i \leftarrow 1$ to $n$ do
        $\widehat{X_i} \longleftarrow KNN(i, X)$ ;            /* Compute $k$ nearest neighbor for $i$ */
        while $N \neq 0$ do
            $\beta \longleftarrow$ RandomeNumber($1, k$) ;             /* chose one neighbor of $i$ */
            for $j \leftarrow 1$ to $p$ do
                $\alpha \longleftarrow$ RandomeNumber($0, 1$);
                $S_{ij} \longleftarrow X_{ij} + (\widehat{X_{i\beta j}} - X_{ij}) \times \alpha$;
            end
            $N \longleftarrow N - 1$;
        end
    end
    return $S$
end
```

and its pseudo code is Algorithm 2 below.

Algorithm 2: SMOTE with Categorical Variables

```
Input: $T$:Training set; $N$ : $N\%$ amount of synthetic samples ; $k$: Number of nearest
       neighbors.
Output: $S$: Set of synthetic samples
begin
    $X \longleftarrow$ MinorityInstances(T);
    $n \longleftarrow$ NumberOfInstances(X);
    $p \longleftarrow$ NumberOfVariables(X);
    $S \longleftarrow \emptyset$;
    if $N < 100$ then
        $n \longleftarrow (N/100) \times n$ ;
        $X \longleftarrow$ RandomSample$(X, m)$;
        $N \longleftarrow 100$;
    end
    $N \longleftarrow N/100$;
    for $i \leftarrow 1$ to $n$ do
        $\widehat{X_i} \longleftarrow KNN(i, X)$ while $N \neq 0$ do
            $\beta \longleftarrow$ RandomeNumber$(1, k)$ ;
            for $j \leftarrow 1$ to $p$ do
                if variable $j$ is categarical variable then
                    $S_{ij} \longleftarrow MajorityVote(\widehat{X_{ij}})$ ; /* compute categorical variable */
                else
                    $\alpha \longleftarrow$ RandomeNumber$(0, 1)$;
                    $S_{ij} \longleftarrow X_{ij} + (\widehat{X_{i\beta j}} - X_{ij}) \times \alpha$;
                end
            end
            $N \longleftarrow N - 1$;
        end
    end
    return $S$
end
```

3.2 Instance Selection

Given a training dataset, instance selection algorithm proposed allows classifiers to achieve the same performance by only using the subset of the training dataset. The algorithm starts with several candidate subsets including instances from the original training dataset. The first candidate subset combines other desired candidate subsets by each iteration until combining that candidate subset cannot further improve the performance of classifiers.

To construct final training dataset, each instance from original detests considered for inclusion in a greedy manner. Specifically, one instance is selected into the final training dataset if including it improve predictive accuracy of the classifier. According to the manner, the instance considered earlier should be given a larger chance to belong to the final training dataset than the later ones. Hence, some of instances considered so late probably never get chance to be selected, and the order really matters. In order to conquer this situation, we create one candidate subset for each instance initially. Totally, we have n candidate subsets and every instance is included in their own index candidate subset. Then we random sample instances for each candidate subset until the size of each candidate subset meets the specified parameter g. By doing that, each instance is selected in at least one candidate subset.

Processing imbalanced datasets with instance selection, we have another concern, that is, regular accuracy usually highly score a classifier which focuses on majority class and underestimates, sometimes even ignores minority class. Instead, we implemented Class Balance Accuracy to evaluate the performance of classifier from candidate subset in each iteration. It is simple to calculate, minority-sensitive, and reflect the performance of a classifier with a single number. After a classifier performs, we can generate confusion matrix C based on the prediction result, where each column of the matrix represents the instances in a predicated class, while each represents the instances in an actual class. A typical confusion matrix as seen in Table 3.1 is a table with two rows and two columns that reports the number of false positives (FP) , false negatives (FN), true positives (TP), and true negatives (TN).

Here, the positive class represents minority class, while the negative class represents major-

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative

Table 3.1: Confusion Matrix

ity class. The Class Balance Accuracy is defined as

$$CBA = \frac{\sum_{i=1}^{k} \frac{C_{ii}}{max(C_{i.}, C_{.i})}}{k} \tag{3.2}$$

where $C_{i.} = \sum_{j=1}^{k} C_{ij}$ and $C_{.i} = \sum_{i=1}^{k} C_{ji}$.

Thus, the instance selection we implemented is algorithm 3 whose pseudo code is proposed below. Note that after fitting the imbalanced dataset with the predefined classifier f, make prediction and further calculate the accuracy based on the same training dataset for the classifier. It could use a test dataset that would be separated before, but using training dataset works fine because the limited number of minority instances, and the test accuracy would barely increase in each iteration.

Algorithm 3: Instance Selection

```
Input: $T$:Training set; $f$: A classifier; $g$: Size of candidate subset.
Output: $S$: Set of selected samples
begin
    $n \longleftarrow$ NumberOfInstances($T$);
    $\widehat{S} \longleftarrow$ CreateCandidateSubsets($n, g$) ;        /* $n$ candidate subsets of $g$ size */
    $\widehat{S} \longleftarrow$ InitialInstances($T$) ;                  /* initially assigne instances */
    for $i \leftarrow 1$ to $n$ do
        $\widehat{S}_i \longleftarrow$ RandomSample($T, g - 1$);
    end
    $S \longleftarrow \emptyset$;
    $currentAccuracy \longleftarrow 0$;
    for $i \leftarrow 1$ to $n$ do
        $fit \longleftarrow$ FitModel($\widehat{S}_i, f$);
        $predicted \longleftarrow$ MakePredict($\widehat{S}_i, fit$);
        $accuracy \longleftarrow$ ClassBalanceAccuracy($\widehat{S}_i, predicted$);
        if $accuracy > currentAccurary$ then
            $S \longleftarrow S \cup \widehat{S}_i$;
            $currentAccuracy \longleftarrow accuracy$;
        end
    end
    return $S$
end
```

3.3 Hybrid Classification Approach of SMOTE and Instance Selection

The proposed hybrid classification approach is inspired by SMOTE and Instance Selection, and constitute of two stages. On the first stage, extract the minority instances from the original dataset and take advantage of SMOTE method to increase the number of minority instances in order to break the decision boundary of the majority class. On the second stage, using greedy selection selects the representative instances from the combined dataset as the final training dataset. After the prepossess, a regular classifier that fits the model by the final dataset outperforms the other classifiers which use those two methods individually, at least has the well performance as those two.

It is no doubt that SMOTE and instance selection are powerful methods to deal with the imbalanced datasets. However, they have some drawbacks when implemented in varied applications. Because of the curse of high dimensions, instance selection is not a appropriate method for non-parameter classifiers if the number of predictors is large. Mechanism behind instance selection is to select the representative instances that are near the decision boundary between the two classes. Those are the key points to really separate the two classes, called support points in support vector machines. Nevertheless, as the number of predictor increase, the number of support points grows up, that means the points that near the decision boundary are probably not near it anymore, and the set of selected instances is supposed to include increasing number of instances in order to conquer the high dimensional scenarios. On the other hand, as the number of minority instances is limited, and instance selection usually selects less than ten percent instances from the original dataset, it is so hard to say the selected instances are enough for a classifier to fit a appropriate model in order to detect the minority class.

SMOTE (Chawla et al., 2002) initially proposed is usually working with under-sampling the majority instances to get real balanced dataset as the final training dataset. By doing that, it weaken the decision region of majority class, and allows a generic classifier to focus more attention on the minority instances. However, the penalty is to miss classified the instances of majority class and over fit the model for minority class. Even the number of instances in training dataset is increased, the overall performance of a classifier is probably not increase as well because a majority of instances generated is duplicated.

The proposed approach is to flourish those two method and ease up their disadvantages by modifying and combining the two. Given an imbalanced dataset, it is very significant to let a classifier to be sensitive to the minority class. In order to make the characteristics and features of minority class outstanding, oversampling minority class is an appropriate idea, as cannot really grab anymore instance from the true density. However, oversampling with replacement is just to make the decision region of minority class much more specific, that not really increase the sensitiveness of a classifier. Instead, the fitted model only detects the specific decision region and even miss classifies other general decision region of minority class. Hence, SMOTE conquers this issue by randomly multiplying a parameter to the feature vector of over sampled instances in order to change the feature spaces of the dataset, rather than modifying the data space. That allows the minority class breaks their specific decision region and invades the majority class border. We implemented a variant of SMOTE approach, Algorithm 2, to out stand the minority class, because the original SMOTE is not able to deal with the categorical variables. On the other hand, we did not under-sampling the instances of majority class which is an importance step for SMOTE, because on the second stage, instance selection would perform sort of step by offering a set of optimized instances from both class. Hence, we extracted only the minority instances from the original dataset and implemented adjusted SMOTE approach to increase the number of minority instances so as to broad the decision region of minority class. For the purpose of avoiding fitted model that overfit the minority class, we kept a certain degree of unbalanced when SMOTEd the minority instances. It is OK to do so, since if keeping very balance is good for classifying those two class, implementation of instance selection latter would take care of it and make them more balance when selecting the best representative instances as final training dataset.

As discussed in previous descriptions, the number of instances of minority class is so limited that most classifiers just consider them as noise, simply classify entire dataset into majority class, and get high score of accuracy if regular accuracy is used as evaluation system. That is part of reason why an evaluation system is one of the two critical parts for implementation of instance selection. We implemented Class Balanced Accuracy as the evaluation system that is very sensitive to the minority class. To interpret the mechanism of Class Balance Accuracy, one can just consider it as regular accuracy but puts more weights on minority class. For example, given 100 points needed to be classified, 10 minority and 90 majority, a regular classifier could only score 50 percent accuracy from Class Balance Accuracy if it classifies correctly all 90 majority class and miss classifies those 10 minority class. On the other hand, another classic learner could get 65 percent accuracy if it correctly classifies 5 instances from minority class and 80 instances from majority class. Another critical part when implementing instance selection is to choose an appropriate classifier or learner. The selected classifier is depended on the specific dataset. As instance selection typically selects the instances that are best representatives of the two classes, and creation of ideal decision boundary effects the accuracy of a generic classifier, the best representative instances should come from the near neighbor of decision boundary. However, the selected instances may be far away from the near neighborhood of decision boundary in p-dimensional space when p is large, leading to a poor prediction and a poor fit. As a general rule, parametric methods will tend to outperform non-parametric approaches when there is a small number of observations per predictor. After SMOTEd stage, the size of dataset has been increased by combining those synthetic minority instances, picking a non-parametric classifier as the assess classifier offers a set of higher quality instance for a generic classifier.

The pseudo code for the proposed combination approach is offered below, Algorithm 4. Note that even though there is no a parameter for imbalanced ratio, it is controlled by adjusting the value of N. There is a boolean variable assignment in line 5 of the Algorithm 4 which decided whether under-sampling the majority class or not. In line 8, we implemented evaluation system as Class Balance Accuracy, but it could be changed by client and various evaluation system could be chosen and switched in order to get high quality instances for final training dataset.

Algorithm 4: Combination Approach Based on SMOTE and Instance Selection

```
Input: $T$:Training set; $f$: A classifier; $g$: Size of candidate subset.
$N$ : $N$% amount of synthetic samples ; $k$: Number of nearest neighbors.
Output: $S$: Set of selected samples
begin
    $S \longleftarrow \emptyset$;
    $X_1 \longleftarrow$ MinorityInstances(T);
    $X_2 \longleftarrow$ MajorityInstances(T);
    $underMaj \longleftarrow$ False;
    $S \longleftarrow SMOTE(X_1, N, underMaj)$;                    /* SMOTE */
    $S \longleftarrow S \cup X_2$;
    $Evaluate \longleftarrow$ ClassBalanceAccuracy;
    $S \longleftarrow InstanceSelection(S, f, Evaluate, g)$;    /* Instance Selection */
    return $S$
end
```

CHAPTER 4. RESULTS

We utilized one standard classifier decision tree to test the performance for our experiments. We experienced on 4 well-known imbalanced data set in UCI datasets and one medical dataset. The four UCI datasets are *car* (car3), *segmentation* (seg1), *yeast* (yeast5), *glass* (g7). The number in the parentheses indicates the target class we choose. Table 4.1 shows the basic characteristics of these five datasets organized based on their negative-to-positive training-instance ratios.

Datase	**# Attributes**	**# Positive**	**# Negative**
g7	10	29	185
seg1	19	30	180
car3	6	69	1659
yeast5	8	51	1433
medical	7	208	2120

Table 4.1: UCI and Medical Datasets

In each dataset, we used K-fold cross validation to evaluate the performance of classifier. By rule of thumb, K is equal to the minimum between square root of n and 5, where n is the number of instances in the dataset. After dividing the dataset into K fold, hold one of them as the *holdout* or testing dataset, the rest of dataset generates training dataset. In order to compare the performances, that training dataset are processed by four different procedures: classic SMOTE, instance selection, hybrid, and control training set. After fitting the decision tree model individually, we evaluate the performances of fits by predicting on the identical testing dataset. We utilized regular accuracy, Area Under Curve (AUC), and Class Balance Accuracy (CBA) to compare the performance of those four strategies.

4.1 SMOTEd Training Dataset

Firstly, we use regular SMOTE to preprocess the training dataset. However, there are two parameters we need to decide, the percentage of over-sampling and percentage of under-sampling. Usually the percentage of over-sampling is a multiple of 100 and the number of generated synthetic minority instances is the multiple of current minority instances in training dataset. On the other hand, the remaining majority instances after under-sampling is the multiple of number of generated synthetic minority instances, and the percentage of under-sampling is also a multiple of 100. For example, given a training dataset with 10 minority instances and 100 majority instances, we set both percentages equal to 200. Then, 20 minority instances will be generated and 40 majority instances are randomly selected. In order to pick the optimal SMOTEd training data set, we utilized the K-fold cross-validation to select the best percentage of over-sampling. The minority class was over-sampled at percentages from 100% to 1000%, while the percentage of under-sampling retains 200 as it depends on the number of over-sampled minority instances. The optimal setting of SMOTE should be related with the percentage of over-sampling with the averaged large of AUC and high accuracy.

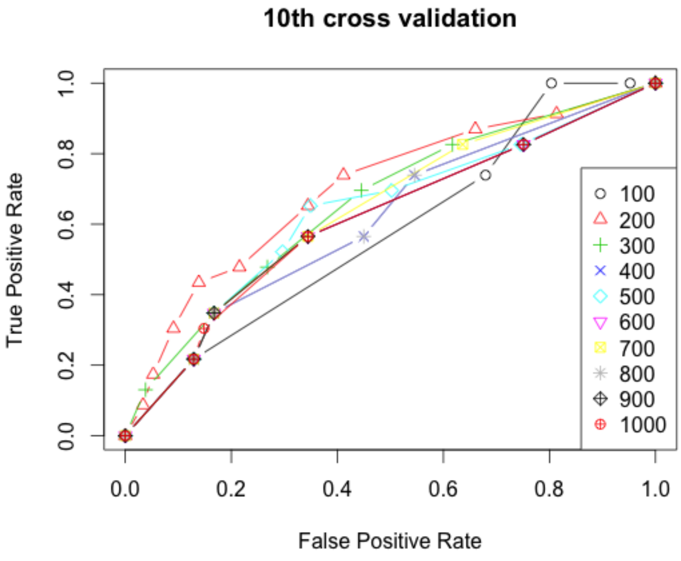

Figure 4.1: ROC Curve of 10th fold of Yearst5

Figure 4.1 is an example of 10th cross validation ROC curve for *yearst5* dataset. Based on this curve, the SMOTE with 200 percent oversampling works best. We can pick the optimum from the boxplot of AUC cross all percentages of oversampling.

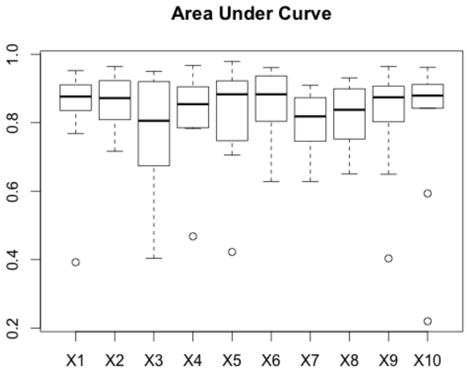

Figure 4.2: All AUC of Yearst5

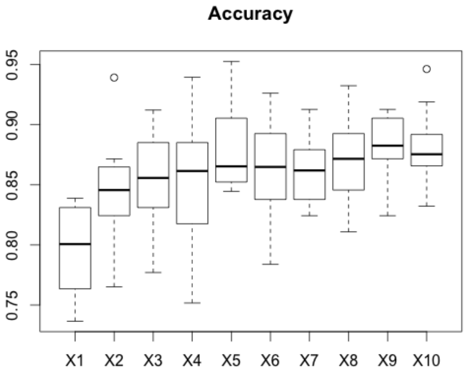

Figure 4.3: All Accuracy of Yearst 5

In Figure 4.2, the optimum SMOTE should be within 100%, 200%, 500%, and 1000%. On the other hand, it would be better if the selected SMOTE also has good performance on regular accuracy. Hence, we also drawn a boxplot for regular accuracy. Compare with those two figures, 4.2 and 4.3, SMOTEs with 100% have good performance on AUC, but it is also the worst on regular accuracy because overfit the minority instances. The exact averaged accuracy and AUC values for those SMOTE are also listed in Table 4.2. In addition to metrics, computational complexity is another significant factor to select the percentage of over-sampling. If the AUC and accuracy are similar, picking a simpler setting for SMOTE would be stable and reduce computing time. Thus, in this case, 200% oversampling is optimum since it has the highest AUC and less complexity, even a little bit lower accuracy.

Metrics	100	200	300	400	500	600	700	800	900	1000
AUC	0.828	**0.866**	0.768	0.822	0.825	0.856	0.805	0.819	0.809	0.798
Accuracy	0.794	**0.847**	0.850	0.853	**0.883**	0.863	0.863	0.871	0.882	0.880

Table 4.2: Averaged AUC and Acc of Yearst5

Implemented this strategy to other datasets, Table 4.3 shows all optimum SMOTE cross all datasets.

Datase	% Over Sample	% Under Sample	AUC	Accuracy
g7	600	200	0.944	0.953
seg1	500	200	0.955	0.971
car3	500	200	0.969	0.929
yeast5	200	200	0.866	0.847
medical	200	200	0.631	0.784

Table 4.3: Optimum SMOTE of Datasets

Note that for datasets *g7* and *seg1*, the best SMOTE oversampling percentage is 600% and 500 % respectively. Since they have 29 and 30 instances from minority class, 174 and 150 synthetic minority instances will be generated and 348 and 300 majority instances will be selected from training dataset. However, there are only 185 and 180 majority instances from *g7* and *seg1*. Thus, what the SMOTE does is select the required number of majority instances with replacement. This does not heart SMOTE. On the other hand, it makes the better performance of decision tree as they are the optimum from those datasets. Understanding the SMOTE behaviors is simple. Minority class encroaches the decision boundary by generating synthetic minority samples, while as response the majority class reinforce the decision boundary though oversampling with replacement. As result, the decision boundary between those two classes becomes clearer and easy to detect by decision tree.

4.2 Instance-Selected Training Dataset

This is pretty simple and straightforward. The nested assess metric was selected to use CBA since it's simple and sensitive to minority class. In addition, we divided the dataset into 100parts and each part was considered as subset candidate. As result, each subset candidate includes 1% data from the whole training dataset. The setting is not optimum. The optimum setting would be set the number of subset candidates equal to n, where n is the number of instances in training dataset. In addition to have equal instances number of subset candidates, optimum setting needs to assign every instance to its index-identical subset candidate in order to make sure every instance is included at least one subset. However, the optimum setting is really time consuming and in most case would not really improve the performance of classifier a lot. Thus, we decided to simplify the strategy and the results showed it worked very well in most case and reduced time complexity a lot. After decision of subset candidates, the instances in each subset will be combined if combination of those instances increases the performance of the classifier. This process go among all subset candidates and output the selected instances as the final training dataset.

4.3 Hybrid Selected Training Dataset

Hybrid has two stages of processing the training dataset by introducing and combining the synthetic minority instances and select the best subset of training dataset from the first stage. For selection and combination of the best subset, the strategy is the same as the Instance Selection, dividing dataset into 100 subset and combining those subset if combination improves the performance of classifier.

In addition to select best subsets, generation of appropriate number of minority class is another significant part of Hybrid method. Because the imbalanced ratios and structures of each dataset are different, there is no consistent oversampling percentage for minority class. Hybrid method usually saves all majority class for further selection. On anther word, the percentage of under-sampling is fixed. So finding an appropriate percentage of oversampling minority class is very important. We utilized the same strategy as finding the optimum of SMOTE. As we could generate very balanced dataset by oversampling minority class, we extended the level of oversampling minority class. Besides oversampling percentage level from 100% to 1000%, we added oversampling levels over 1000% until the dataset becomes very balance. For example, there are 5 minority class and 100 majority class. For finding the optimum of SMOTE, we set up oversampling minority class from 100% to 1000%. For finding the optimum of hybrid, oversampling levels from 1100% to 2000% are also included since the dataset is very balance if percentage of oversampling is chosen to be 2000%. In order to find the optimum hybrid, the same assess metrics that were used for best SMOTE were also implemented for best hybrid, that is, AUC and accuracy.

Figures 4.4 and 4.5 show that AUC plot is fluctuated but has a increasing trend. In order to pick up an appropriate point, comprehensive consideration about means and variances of each

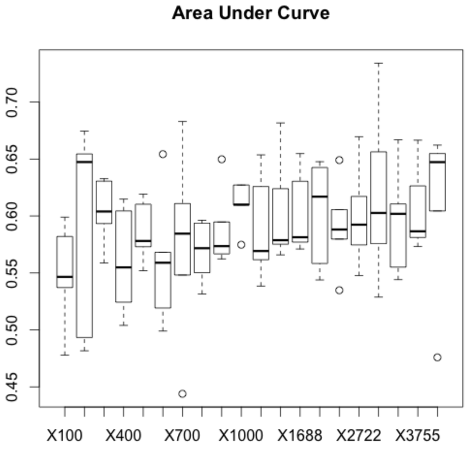

Figure 4.4: All AUC of Hybrid-based Medical Dataset

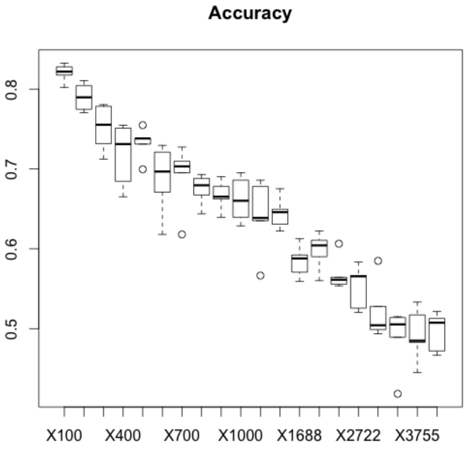

Figure 4.5: All Accuracy of Hybrid-based Medical Dataset

level oversampling is essential. Plus, accuracy plot, Figure 4.5, is monotonically decreasing. It seems there is a trade-off between AUC and Accuracy. That does make sense because based on ROC curve itself, it is the mechanism for any classifier that the higher true positive rate, and the higher false positive rate. In anther word, the more classifier identifies the minority class, the more misclassifies the majority class. Model complexity is another essential part needed to be considered. We usually pick up the simpler one if both settings have the almost identical AUC and Accuracy. For instance, in this case, we selected 1000% as percentage for oversampling minority class, because it has very both high AUC and accuracy, and very stable comparing with other settings. Based on the strategy to select the optimum of hybrid method, all optimal hybrids are listed on Table 4.4.

Datase	**% Over Sample**	**AUC**	**Accuracy**
g7	1600	0.931	0.949
seg1	500	0.864	0.867
car3	900	0.971	0.939
yeast5	3300	0.885	0.881
medical	1900	0.610	0.662

Table 4.4: Optimum Hybrid of Datasets

4.4 Results

The original training dataset has been preprocessed by four different strategies, regular SMOTE, instance selection, hybrid method, and control set. Then we got four different training datasets, utilized the regular classifier, decision tree, to fit each individual training dataset, and predicted the fit onto identical test dataset. We processed this steps over 100 times among each datasets. For each dataset, we are able to evaluate the prediction through computing AUC, CBA and accuracy.

4.4.1 CBA Assess Metric

Firstly, we utilized assess metric as nested metric for instance selection and first stage of hybrid in order to select best subset of training dataset.

g7	**AUC**	**CBA**	**Accuracy**	**# Minority**	**# Majority**
Control	0.917	0.868	0.934	5	148
Selection	0.500	0.607	0.607	0	6
Hybrid	0.924	0.899	0.934	10	7
SMOTE	0.924	0.899	0.934	35	60

Table 4.5: CBA-based Evaluation for g7 dataset

seg1	**AUC**	**CBA**	**Accuracy**	**# Minority**	**# Majority**
Control	0.969	0.930	0.978	30	180
Selection	0.805	0.788	0.902	6	13
Hybrid	0.808	0.765	0.891	7	6
SMOTE	0.964	0.942	0.977	180	300

Table 4.6: CBA-based Evaluation for seg1 dataset

car3	**AUC**	**CBA**	**Accuracy**	**# Minority**	**# Majority**
Control	0.785	0.675	0.856	13	1327
Selection	0.500	0.856	0.856	0	14
Hybrid	0.942	0.761	0.891	9	35
SMOTE	0.963	0.800	0.925	78	130

Table 4.7: CBA-based Evaluation for car3 dataset

yeast5	**AUC**	**CBA**	**Accuracy**	**# Minority**	**# Majority**
Control	0.611	0.492	0.883	10	1146
Selection	0.500	0.875	0.875	0	12
Hybrid	0.766	0.709	0.890	23	49
SMOTE	0.823	0.633	0.842	30	40

Table 4.8: CBA-based Evaluation for yeast5 dataset

medical	**AUC**	**CBA**	**Accuracy**	**# Minority**	**# Majority**
Control	0.567	0.668	0.717	41	1696
Selection	0.502	0.695	0.716	0	22
Hybrid	0.580	0.549	0.658	40	70
SMOTE	0.582	0.503	0.624	123	164

Table 4.9: CBA-based Evaluation for Medical dataset

Based on the Tables from 4.5 to 4.9, best SMOTE method usually works best, while hybrid method is also competitive and has smaller size of training dataset. On the other hand, instance selection fails in many case because the classifier learning from instance-selection-based training dataset classifies almost all instances to majority class and misclassifies all minority class.

4.4.2 AUC Assess Metric

For instance selection based methods, nested assess metric is very important. A sensitive-minority assess metric will be a good helper for a classifier to select most minority class, but not overfit. Since accuracy and CBA both assumes the cutoff value is equal to 0.5, the selected instances are not very comprehensive. However, AUC considers all possible cutoff values. Hence, the selected instances should be more stable and comprehensive.

g7	**AUC**	**CBA**	**Accuracy**	**# Minority**	**# Majority**
Control	0.917	0.868	0.934	5	148
Selection	**0.872**	**0.862**	**0.976**	**7**	**26**
Hybrid	**0.991**	**0.887**	**0.954**	**33**	**15**
SMOTE	0.924	0.899	0.934	35	60

Table 4.10: AUC-Based Evaluation for g7 dataset

Based on the Tables from 4.10 to 4.14, those two instance-select-based approaches, instance

seg1	AUC	CBA	Accuracy	# Minority	# Majority
Control	0.969	0.930	0.978	30	180
Selection	**0.959**	**0.926**	**0.977**	**67**	**15**
Hybrid	**0.965**	**0.929**	**0.975**	**49**	**51**
SMOTE	0.964	0.942	0.977	180	300

Table 4.11: AUC-Based Evaluation for seg1 dataset

car3	AUC	CBA	Accuracy	# Minority	# Majority
Control	0.785	0.675	0.856	13	1327
Selection	**0.764**	**0.856**	**0.856**	**1**	**26**
Hybrid	**0.967**	**0.792**	**0.920**	**17**	**85**
SMOTE	0.963	0.800	0.925	78	130

Table 4.12: AUC-Based Evaluation for car3 dataset

selection and hybrid, with AUC have better results than with CBA. Instance selection was to assign all instance to majority class when it has CBA as nested assess metric. As result, it scores only 0.5 in AUC. In another word, it just randomly guess when assigning an instance. When it applies AUC as nested assess metric, it instead is able to correctly classifier some minority class. In addition, the sizes of subset of instance selection and hybrid are increased. For instance selection, increasing number of minority class make the fit to put more attention on minority class rather than ignoring them. For hybrid approach, increasing numbers are for majority class and some minority class, which make the decision boundary clearer. As result, classifier has better performance. Specifically, hybrid approach allow decision tree has better performance than SMOTE in term of AUC, but uses less number of instances. In addition, Hybrid approach has competitive results in term of CBA and Accuracy. Thus, comprehensive assess metric like AUC is more appropriate as an assess metric for instance-select-based approach. Furthermore,

yeast5	AUC	CBA	Accuracy	# Minority	# Majority
Control	0.611	0.492	0.883	10	1146
Selection	**0.789**	**0.875**	**0.875**	**1**	**23**
Hybrid	**0.788**	**0.662**	**0.881**	**47**	**115**
SMOTE	0.823	0.633	0.842	30	40

Table 4.13: AUC-Based Evaluation for yeast5 dataset

medical	**AUC**	**CBA**	**Accuracy**	**# Minority**	**# Majority**
Control	0.567	0.668	0.717	41	1696
Selection	**0.562**	**0.658**	**0.714**	**5**	**78**
Hybrid	**0.592**	**0.521**	**0.657**	**71**	**119**
SMOTE	0.582	0.502	0.624	123	164

Table 4.14: AUC-Based Evaluation for medical dataset

hybrid is very flexible because of its various setting such as percentage of oversampling minority class, different assess metrics, and even different selection strategy.

CHAPTER 5. SUMMARY AND DISCUSSION

We proposed hybrid approach to deal with imbalanced dataset. Hybrid approach is a very robust and promising approach since it has varied settings. The numeric results show that hybrid is able to allow classifier has better performance than SMOTE and instance selection if set up an appropriate assessment metric. In this thesis, we applied AUC and CBA. Furthermore, hybrid approach needs less instances than SMOTE and way faster than greedy selection. For greedy selection, over ninety percent of its computing time is occupied selection of suitable setting such as the number of candidate subset and the size of candidate subset. Instead, we simply separated individual dataset into 100 parts, each part is considered as a candidate subset, and the instances retained in individual candidate are determined as the size of a candidate subset. This simplification saves a lot of computational time but with hybrid it has better performance. Comparing the evaluation of AUC-based and CBA-based, a comprehensive assessment metric works better than non-comprehensive since comprehensive assessment metrics use all of the cutoff value to evaluate, in the mean while non-comprehensive ones only use one cutoff value.

In future research, how to select an appropriate characteristic for hybrid is essential. Cross-validation is a good method to select the ideal parameter like percentage of oversampling minority class. However, filtering an ideal nested assess metric, or even good instance selection approach is harder.

BIBLIOGRAPHY

Batista, G. E., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, 6(1):20–29.

Bezdek, J. C. and Kuncheva, L. I. (2001). Nearest prototype classifier designs: An experimental study. *International Journal of Intelligent Systems*, 16(12):1445–1473.

Cano, J. R., Herrera, F., and Lozano, M. (2003). Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study. *Evolutionary Computation, IEEE Transactions on*, 7(6):561–575.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16(1):321–357.

Chawla, N. V., Lazarevic, A., Hall, L. O., and Bowyer, K. W. (2003). Smoteboost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119. Springer.

Chou, C.-H., Kuo, B.-H., and Chang, F. (2006). The generalized condensed nearest neighbor rule as a data reduction method. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 556–559. IEEE.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27.

Estabrooks, A., Jo, T., and Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549.

He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328. IEEE.

He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284.

Jo, T. and Japkowicz, N. (2004). Class imbalances versus small disjuncts. *ACM SIGKDD Explorations Newsletter*, 6(1):40–49.

Kubat, M., Matwin, S., et al. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *ICML*, volume 97, pages 179–186. Nashville, USA.

Kuncheva, L. I. (1995). Editing for the k-nearest neighbors rule by a genetic algorithm. *Pattern Recognition Letters*, 16(8):809–814.

Kuncheva, L. I. (1997). Fitness functions in editing k-nn reference set by genetic algorithms. *Pattern Recognition*, 30(6):1041–1049.

Ling, C. X. and Li, C. (1998). Data mining for direct marketing: Problems and solutions. In *KDD*, volume 98, pages 73–79.

Lumini, A. and Nanni, L. (2006). A clustering method for automatic biometric template selection. *Pattern Recognition*, 39(3):495–497.

Olvera-López, J. A., Carrasco-Ochoa, J. A., and Martínez-Trinidad, J. F. (2010a). A new fast prototype selection method based on clustering. *Pattern Analysis and Applications*, 13(2):131–141.

Olvera-López, J. A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., and Kittler, J. (2010b). A review of instance selection methods. *Artificial Intelligence Review*, 34(2):133–143.

Raicharoen, T. and Lursinsap, C. (2005). A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (poc-nn) algorithm. *Pattern recognition letters*, 26(10):1554–1567.

Weiss, G. M. and Provost, F. (2001). The effect of class distribution on classifier learning: an empirical study. *Rutgers Univ.*

Wilson, D. R. and Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms. *Machine learning*, 38(3):257–286.

Zhang, H. and Sun, G. (2002). Optimal reference subset selection for nearest neighbor classification by tabu search. *Pattern Recognition*, 35(7):1481–1490.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Sigurdur Olafsson for his guidance, patience and support throughout this research and the writing of this thesis. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members for their efforts and contributions to this work: Dr. Sarah Ryan Tanner and Dr. Dianne Cook. I would additionally like to thank my wife, Xinyao Li, for her help and support. She have been at my side to accompany and support me.