University of California
Los Angeles

Classification of Imbalanced Data Using Synthetic Over-Sampling Techniques

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Statistics

by

Peng Jun Huang

2015

Abstract of the Thesis

Classification of Imbalanced Data Using Synthetic Over-Sampling Techniques

by

Peng Jun Huang

Master of Science in Statistics

University of California, Los Angeles, 2015

Professor Yingnian Wu, Chair

A dataset is considered to be imbalanced if the classification objects are not approximately equally represented. The classification problems of imbalanced dataset have brought growing attention in the recent years. It is a relatively new challenge in both industrial and academic fields because many machine learning techniques do not have a good performance. Often the distribution of the training data may be different than that of the testing data. Typically, sampling methods would be used in imbalanced learning applications, which modifies the distribution of the training samples by some mechanisms in order to obtain a relatively balanced classifier. A novel synthetic sampling technique, SMOTE (Synthetic Minority Over-sampling Technique), has showed a great deal of success in many applications. Soon after this powerful methods was introduced, some other SMOTE-based sampling methods such as SMOTEboost , Border-line SMOTE and ADASYN (Adaptive Synthetic Sampling) have been developed. This paper reviews and compares some of these synthetic sampling methods for learning imbalanced datasets.

The thesis of Peng Jun Huang is approved.

Nicolas Christou

Frederic Paik Schoenberg

Yingnian Wu, Committee Chair

University of California, Los Angeles

2015

To my dear parents . . .

for constantly supporting me through my educational journey

and giving me the most motivation

to succeed in everything I pursue.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Imbalanced dataset, which can be found in real world applications of various fields, can cause extremely negative effect on the performance of machine learning algorithms. These applications range from text classification, bioinformatics, speech recognition and telecommunications. This is a This a new challenge that often causes inaccurate statistical modeling and classification. Imbalance with the order of 100 to 1 is prevalent in fraud detection and imbalance of up to 100,000 to 1 has been considered as other applications [1], [12]. When dealing with imbalanced dataset, we should not use standard machine learning algorithms since it may cause a critical mistake; instead, we may consider classifiers with adjusting the output threshold. In the real world, we usually have far more data than the algorithm can deal with and we have to select and sample, or we have no data at all and we have to create them artificially. In the first situation, we need to think about how much sample we should obtain and what the proportion of major group and minor group should be. In the second situation, we need to consider what the cost of creating samples and how many we should create with what proportion.

Sampling methods can be further classified into oversampling and undersampling. Oversampling is an intuitive method that increases the size of the minority class; on the other hand, undersampling is to use a subset of the majority class to train the classifier. In real world applications, both methods have showed positive effects on the performance of many classifiers. This paper focuses on oversampling techniques on two-class imbalanced learning problems, as well as provides a re-

view of several oversampling approaches, which are random oversampling with replacement, synthetic minority over-sampling (SMOTE) and adaptive synthetic sampling approach. The combinations of these oversampling methods and different classifiers will be applied on various imbalanced dataset to compare and measure their performances.

In order to compare the performances of the combinations of oversampling methods and classifiers, traditional metric such as, overall accuracy, precision and recall, do not give a meaningful evaluation. This is not appropriate because of the natural problem of imbalance. For example, consider an imbalanced dataset with 95% majority class and only 5% minority class. A simply default strategy of guessing the majority class would give a predictive accuracy of 95%. However, the minority class examples are usually the observations that we are interested in, which requires a high correction rate on predicting minority class, with a reasonable tolerance of error rate in the majority class. Therefore, a high overall predictive accuracy does not give great confidence on the classifier. Some typical metrics to measure these methods performance are F-measure and G-measure. Additionally, a standard technique for summarizing classifier performance, the Receiver Operating Characteristic (ROC) curve is also a good metric used in imbalanced situation. ROC curve summarizes over a range of tradeoffs between true positive and false positive error rates. The Area Under the Curve (AUC) is widely used as a measurement of classifiers performance. All these metric will be further discussed in the later chapter and also be calculated in each experiment.

CHAPTER 2

Performance Measure

2.1 Confusion Matrix

Most of the metrics mentioned above can be calculated from the confusion matrix as illustrated in Figure 2.1, which is a typical tools used to record the performance of classifiers.

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

Figure 2.1: Confusion Matrix

The rows of the matrix are the actual class while the columns are the predicted class. In confusion matrix, TN, FN, FP and TP represent true positives (the number of negative samples correctly classified, similar definition for the rest), false negatives, false positives, false negatives and true positive respectively. Because under imbalanced learning condition, a comprehensive assessment of a learning algorithm should not be evaluated by a single metric, but a set of metrics. The next section will introduce a set of assessment metrics, derived from confusion matrix, related to ROC graphs to evaluate the performance of classification approaches.

2.2 Evaluation Metrics

A typical classification evaluation criterion is overall accuracy. This metric provides a comprehensive assessment when the dataset is relatively balanced; however, this is not the case in imbalanced situation. It measures the percentage of the examples that are correctly classified. It is defined as:

$$\textit{Overall Accuracy (OA)} = \frac{TP + TN}{TP + FP + FN + TN}$$

Precision is the fraction of predicted positive examples that are actually positive. Maximum precision (no FP) corresponds to absence of type I error.

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

Recall is the fraction of actual positive examples that are predicted as positive. It is also named specificity. Maximum recall (no FN) corresponds to absence of type II error.

$$\textit{Recall (Sensitivity)} = \frac{TP}{TP + FN}$$

It is more difficult to obtain both high precision and high recall simultaneously in imbalanced learning than in balanced learning. It is common for a classifier to achieve high rate in one of these metrics but low rate in the other. F-measure, also known as F-score or F-measure, is a typical metric for binary classification, which can be interpreted as a weighted average of the precision and recall. The general formula of F-measure is as:

$$F - measure = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

where greater β gives higher weights on recall. In the balanced case, β is set to be one. This yields to the harmonic mean of precision and recall. This F-measure will be one important evaluation metric in all the experiments in this paper.

Another common metric, the G-mean metric, is the geometric mean of true positive rate and true negative rate. The former term is the same as recall, or sensitivity; while the later term is also known as specificity. G-mean evaluates the degree of inductive bias in terms of the ratio of positive accuracy and negative accuracy. It is defined as:

$$Specificity = \frac{TN}{TN + FP}$$

$$G - mean = \sqrt{sensitivity * specificity}$$

Although both F-measure and G-mean are great improvements to overall accuracy, they are not the only metrics being used. Because many performance assessment metrics have been used by different authors, it is difficult to make a comparison among different learning approaches.

2.3 Receiver Operating Characteristics (ROC) Curves

The receiver operating characteristic curve, or ROC curve, is a assessment technique that makes use of the true positive rate and false positives rate, as illustrated in Figure 2.2. On an ROC curve, the x-axis is FP percentage, which can be calculated by the formula, $FP\% = \frac{FP}{FP+TN}$. It is equivalent to $1 - specificity$. The y-axis is TP percentage or recall, which calculated by the formula, $TP\% = \frac{TP}{TP+FN}$. By plotting the TP percentage over FP percentage, ROC curve provides a visual representation of the tradeoff between the benefits (TP%) and the costs (FP%). The points on a ROC curve corresponds to the performance of a single classifier on a given distribution. The line $y = x$ represents the scenario of guessing the class randomly. In imbalanced dataset, the original ratio of minority class to majority class is small, it is hard to detect the positive examples by a classifier; therefore, we tend to obtain low true positive rate and low false positive rate. In this case, the point is usually at the lower left corner. By varying the distribution of the training set, the corresponding point of a classifier will move from one position to

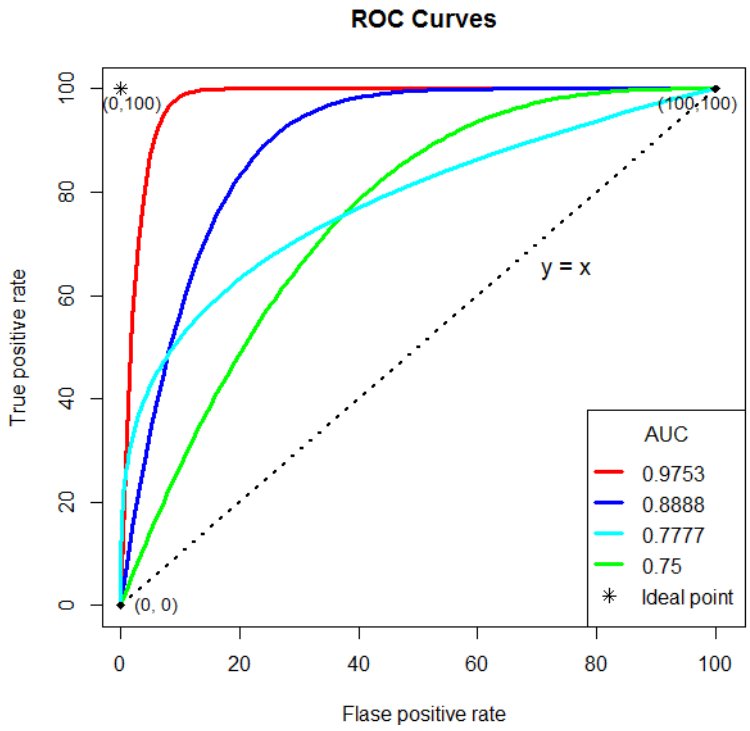

Figure 2.2: As the ratio of minority class to majority class in training set increases, the operation point will move to the upper right.

another along the curve. We can change the distribution of the training set by oversampling the minority class or undersampling the majority class.

Our target is to achieve high true positive rate and low false positive rate for a given classification learning, the ideal point appears at the upper left corner (0, 100). Although this point is almost impossible to achieve, we should try to develop a method to obtain a curve as closer to the ideal point as possible. This brings us to a question: how can we determine which classification learning method outperforms than the others if the ROC curves are intersecting each other?

The Area Under the ROC Curve (AUC) gives the comprehensive solution to this question. This metric is a standard technique used to compare the performances of classification learners. As illustrated in Figure 2.2, it is difficult to select be-

tween the cyan and the green classifier based on the ROC curves; however, AUC quantifies their overall performances given different class distributions. By comparing AUC, we can establish a dominance relationship between classifiers.

Several evaluation metrics discussed above are used in all experiments throughout this paper. A classification approach could be shown that it is superior under one metric while another metric might indicate it is less desirable than other approaches. Therefore, evaluation should be considered under different metrics and one can select an appropriate metric based on his or her knowledge of the application. The next chapter will discuss several efficient oversampling techniques that have been widely applied in many real-world imbalanced dataset.

CHAPTER 3

Oversampling Methods

The nature problem of imbalanced learning is the extreme ratio of minority class and majority class cause biases on making decision for a classifier. In this situation, most of minority instances could be easily classified into majority group, causing the detection of minority instance difficult. Many evaluation metrics suggest that classifiers perform poorly, detection rate of minority class is low. However, this does not mean that classifiers can't learn from imbalanced dataset. The use of sampling methods on imbalanced learning is to modify the dataset by some mechanisms in a way that they can achieve a more balanced distribution. Oversampling and undersampling act as a preprocessing phase, but this paper only discusses oversampling. Several famous sampling methods, random oversampling with replacement, synthetic minority oversampling technique (SMOTE), adaptive synthetic sampling technique (ADASYN), will be discussed in this chapter.

3.1 Random Oversampling with Replacement

Random oversampling with replacement is adding a set E sampled from the minority class by the following mechanics: for a set of randomly selected minority examples in S_{min}, augment the original set S by replicating the selected examples and adding them to S [8]. By doing this, the size of total examples in S_{min} will be increased by $|E|$ and the class distribution becomes more balanced accordingly. This inquisitive method simply replicates a portion of minority class to increase the weights of those examples. Because the replacement process is totally random,

this method does not specify a clear borderline of two classes. However, because this method re-creates some existing examples in the original minority class, its main drawback is that it could cause overfitting [4]. This methods is the fundamental concept of oversampling technique. Many other common oversampling algorithms used in real-world applications are developed based on this method.

3.2 Synthetic Minority Oversampling Technique (SMOTE)

Inspired by a technique that has successfully proved in handwritten character rcognition [5], [10], Synthetic Minority Oversampling Technique (SMOTE) was proposed by Chawla in 2002 [1]. Unlike random oversampling, in SMOTE algorithm minority class is oversampled by generating synthetic examples rather than by oversampling with replacement. The SMOTE algorithm creates artificial examples based on the feature space, rather than data space, similarities between existing minority examples [1] [8]. These synthetic examples are generated along the line segments joining a portion or all of the K nearest neighbors of the minority class. Depending on the amount of the oversmapling required, neighbors from the K nearest neighbors are randomly chosen.

More specially, let $S_{min} \in S$ represent the minority class. For each example $x_i \in S_{min}$, find the K-nearest neighbors, given an specified K. The K-nearest neighbors are defined as the K elements of S_{min} whose euclidian distance between itself and x_i have the smallest magnitude in the feature space X. To create a new sample, select one of the K-nearest neighbors randomly, and then find the difference between the selected sample and its nearest neighbor. Multiply this difference by a number generated uniformly from 0 to 1; however, one might modify this factor by changing uniform distribution to other distribution depending on the application. Finally, add this vector to the selected sample x_i

$$x_{syn} = x_i + (\hat{x}_i - x_i) \times \lambda,$$

where $x_i \in S_{min}$ is the selected instance from minority class, $\hat{x}_i \in S_{min}$ is one of the K-nearest neighbors of x_i and $\lambda \in [0, 1]$ is random generated number.

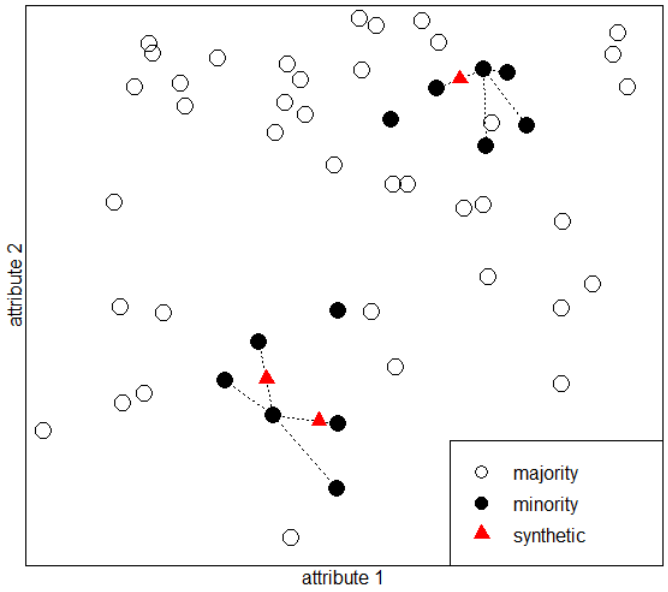

Figure 3.1: Synthetic Minority Over-sampling Technique with K = 4

Figure 3.1 demonstrates an example of SMOTE procedure. The number of K-nearest neighbors is set to be $K = 4$. The figure shows the synthetic examples are created along the line segment between x_i and $\hat{x}_i$ in red. Theses synthetic examples help balance the original class distribution, which generally significantly improves learning. However, SMOTE algorithm also has its disadvantage such as over generalization of the minority class space. From the original SMOTE algorithm, many other SMOTE-based algorithms have been developed over the years and some of them effectively improve the performance on imbalanced learning.

Algorithm 1 SMOTE(X, N, K)

Input:

X: the original training set

N percentage of oversampling

K: number of nearest neighbors

Output: the oversampled training set

$n \leftarrow$ # observations

$m \leftarrow$ # attributes

$n_{min} \leftarrow$ # min observations

if $N < 100$ **then**

Stop: warning "N should be greater than 100"

end if

$N \leftarrow int(N/100)$

$S_{(n*N)\times m} \leftarrow$ empty array for synesthetic samples

for $i \leftarrow 1$ to n_{min} **do**

for each i, compute k nearest neighbors and store the indices in the nn

$newindex \leftarrow 1$

while $N \neq 0$ **do**

$K_c \leftarrow$ random number between 1 and K

for $j \leftarrow 1$ to m **do**

$diff \leftarrow= X[nn[K_c]][j] - X[i][j]$

$gap \leftarrow uniform(0, 1)$

$synthetic[newindex][j] \leftarrow X[i][j] + gap \times diff$

end for

$newindex+ = 1$

$N- = 1$

end while

end for

Return *Dataframe (X & synthetic)*

3.3 Adaptive Synthetic Sampling Approach (ADASYN)

Based on the idea of SMOTE, various oversampling methods have been introduced later on such as SMOTEBoost, Borderline-SMOTE [6] and Adaptive Synthetic Sampling (ADASYN), which all have been shown improvement on imbalanced learning on different dataset. Of particular interest with these adaptive algorithms are the techniques used to identify minority examples.

Before discussing ADASYN, lets consider the case of Borderline-SMOTE, which provides the basic idea of the development of ADASYN. This algorithm is achieved as the following. First, for each $x_i \in S_{min}$ determine the set of nearest neighbors, denoted as $S_{i:m}$, where m is the number of minority class, and determine the number of nearest neighbors that belongs to the majority class, i.e., $|S_{i:m} \cap S_{maj}|$. Then it classifies each x_i into three groups, "Danger", "Safe", and "Noise", based on the number of majority examples in its K-nearest neighbors. x_i belongs to "Danger" if:

$$\frac{m}{2} \leq |S_{1:m} \cap S_{maj}| < m$$

These examples represent the borderline minority class examples. If $|S_{1:m} \cap S_{maj}| = m$, i.e., all the K nearest neighbors of x_i are majority examples, then x_i is considered as "Noise"; otherwise, it is "Safe". Unlike SMOTE, Borderline-SMOTE only creates synthetic examples for those near the border, while no synthetic examples should be generated for "Noise" instances.

On the other hand, ADASYN, which adapts the concept of Borderline-SMOTE, creates different amount of synthetic examples for minority class according to their distribution. ADASYN algorithm decides the number of synthetic examples that need to be generated for each minority example by the amount of its majority nearest neighbors [7]. The more majority nearest neighbor, the more synthetic examples will be created.

More specifically, the total number of synthetic data examples, G, is determined by $\beta \in [0, 1]$, which is a parameter used to specifiy the balanced level after the synthetic process.

$$G = (|S_{maj}| - |S_{min}|) \times \beta$$

Then find the K-nearest neighbors for each example $x_i \in S_{min}$. Define the weight of x_i for synthetic process as the density distribution Γ_i:

$$\Gamma_i = \frac{\Delta_i / K}{Z}, \quad i = 1, ..., |S_{min}|$$

where Δ_i is the number of majority examples in the K-nearest neighbors of x_i, Z is a normalized constant so that Γ_i is a probability mass function, i.e., $\sum \Gamma_i = 1$. Then determine the number of synthetic examples that need to be generated,g_i for each $x_i \in S_{min}$:

$$g_i = \Gamma_i \times G$$

Finally, using SMOTE algorithm to generate g_i synthetic examples for each $x_i \in S_{min}$. ADASYN was developed based on the idea of SMOTE algorithm. The difference is that ADASYN use density distribution Γ as a criterion to decide the number of synthetic examples; while in SMOTE, each minority examples has equally likely chance to be selected for synthetic process. Unlike Borderline-SMOTE, ADASYN does not identify "Noise" instances so it is possible to create a large amount of artificial data around those instances, which may create an unrealistic minority space for the learner.

Algorithm 2 ADASYN(X, β, K)

Input:

X: the original training set

$\beta \in [0, 1]$: desired balanced level

K: number of nearest neighbors

Output: the oversampled training set

S_{maj} majority class S_{min} minority class

n_{maj} # of majority observations n_{min} # of minority observations

$G \leftarrow (n_{maj} - n_{min}) \times \beta$

$r_{1 \times n_{min}}$ percentage of nearest neighbors in majority class

for $i \leftarrow 1$ to n_{min} **do**

for each i, compute k nearest neighbors and store the indices in the nn

$r[i] \leftarrow \frac{|nn[i] \cap S_{maj}|}{K}$

for $i \leftarrow 1$ to n_{min} **do**

$\hat{r}[i] \leftarrow \frac{r[i]}{\sum_i r[i]}$

$g[i] \leftarrow int(\hat{r}[i] \times G)$

$Syn_{(G) \times m} \leftarrow$ empty array for synesthetic samples

$j = 1$

for $i \leftarrow 1$ to n_{min} **do**

$newindex \leftarrow g[i]$

while $newindex \neq 0$ **do**

$K_c \leftarrow$ random number between 1 and K

$diff \leftarrow= S_{min}[nn[i][K_c]] - S_{min}[i]$

$Syn[j][] \leftarrow S_{min}[i][] + diff \times uniform(0, 1)$

$j+ = 1$

$n- = 1$

end while

end for

Return *Dataframe (X & Syn)*

3.4 Other Oversampling Techniques

Besides ADASYN algorithm, multiple modification have been proposed based on the idea of the original SMOTE algorithm. SMOTE-NC (Synthetic Minority Oversampling Technique Nominal Continuous) and SMOTE-N (Synthetic Minority Oversampling Technique Nominal)are proposed by Chawla later which are the extended version of SMOTE handling datasets with nominal features [1] . SMOTEBoost has also been proposed which changes the updating weights and compensating for skewness [2]. Exceptional improvement of its performance has been shown in F-measure.

Proposed by Hui Han and Wen-Yuan Wang [6], borderline-SMOTE1 and borderline-SMOTE2 are two new oversampling techniques. In these two algorithms, only the minority examples close to the borderline are being over-sampled, which adaptively creates a clearer separation of two classes. Their paper has shown that these two algorithms achieve better F-value and TP rate than SMOTE. Borderline oversampling has been used widely for imbalanced data classification. [11]

Data extracted from medical images usually resulted in geometric complexity in data classification. Additionally, they can not be separated linearly in Euclidean space. In order to deal with this problem, Juanjuan Wang et al have proposed an novel method that improves SMOTE algorithm by incorporating LLE algorithm (locally linear embedding algorithm)[14]. This approach maps the high-dimensional data into a low-dimensional space where the data can be oversampled by SMOTE.

Support vector machine has been widely used for the classification of imbalanced dataset. Several revised versions of SVM that combines over and under sampling techniques such as SVMs, SVM-C, SVM-SMOTE and SVM-RU, have been presented [13], [15]. A cluster-based oversampling algorithm dealing with between-

class imbalance and within-class imbalance simultaneously has been proposed by Taeho Jo et al [9]. A multiple re-sampling method that selects an appropriate re-sampling rate adaptively has been proposed by Andrew Estabrooks [3].

Many other oversampling methods have proposed for solving imbalanced problems. Many other methods such as undersampling, cost-sensitive learning and hybrid algorithm have been prosed in real-world applications from various fields over years. However, we don't emphasize them in this paper.

CHAPTER 4

Experiments

4.1 Dataset description

We compared different oversampling techniques on various real world imbalanced machine learning datasets. The basic summary of these datasets sorted by imbalance ratio are as shown in Table 4.1. All datasets are accessible on UCI Machine Learning Repository. Because we only focus on two-class imbalanced classification problems, some of the original datasets need to be modified in preprocess stage according to different literary result from similar experiments. A brief description of modification is discussed in the following.

Dataset name	total examples	minority examples	majority examples	atrributes	imbalance ratio
ionosphere	351	126	225	34	0.56
glass1	214	76	138	9	0.551
PID	768	268	500	8	0.536
vehicle	846	199	647	18	0.308
vowel	990	90	900	10	0.1
glass2	214	17	197	9	0.086
abalone	731	42	689	7	0.061

Table 4.1: Dataset description

Ionosphere dataset: This dataset contains 351 observations with a binary response, 225 good radar returns and 126 bad radar returns. The bad radar returns

are treated as minority class. There are 34 numerical attributes, all of which will be used.

Glass1 dataset: This dataset contains 214 observations with a binary response, 76 positive and 138 negative. The positive examples are treated as minority class. There are 9 continuously numerical attributes, all of which will be used.

Pima India Diabetes dataset: This dataset contains 768 females at least 21 years old of Pima India heritage. The response variable is binary, diabetic cases (positive) and nondiabetic cases (negative). There are 268 positives and 500 negatives. All 8 attributes are numerical and all of them will be used for analysis.

Vehicle dataset: This dataset has a total of 846 examples with 4 type of vehicles, opel, saab, bus and van. It is used to classify a given silhouette. Each example is represented by 18 attributes. Because we are only interested in two-class classification, Van has been chosen as the minority class and combine the remaining classes into a majority class. After the modification, the dataset become imbalanced with 199 minority examples and 647 majority examples.

Vowel recognition dataset: This dataset is used to classify different vowels related to speech recognition. The data contains 990 examples with 13 attributes. The first 3 attributes are grouping and identifiers so they are moved, as a result of 10 attributes for analysis. The response variable has 11 classes in the original dataset and we only choose the first vowel as the minority, giving us 90 and 900 minority and majority class examples, respectively.

Glass2 dataset: This dataset is the same as glass1 dataset except that it has a different imbalance ratio. There are also 214 observations and 9 attributes, but only with 17 minority class examples. This is a dataset with imbalance ratio about 1:12.

Abalone dataset: This dataset is used to predict the age of abalone from physical

measurements. It originally contains 4177 observations and 8 attributes. There are 29 classes, and we choose the class "18" as the minority class and class "9" as the majority class as suggested in another research paper. Moreover, the attribute "sex" is removed for analysis. This gives 42 and 689 minority and majority class examples respectively with a total of 7 numerical attributes.

4.2 Experiment results

In our experiments, four classifiers, naive bayes, decision tree, random forest, and supported vector machine, are used as the learning models, combined with several oversampling techniques, oversampling with replacement, SMOTE, and ADASYN. As a reference, we also provide the performance of those classifiers based on the original imbalanced dataset without using any oversampling methods. The assessment metrics presented in Chapter 2 are used to illustrates the performance of these learning techniques. The results are based on the average of 100 runs and at each run, half of the minority and majority class examples are used for training purpose and the rest are used for testing. All seven datasets discussed above are used for the experiment under the same learning methods. We will compare different learning methods within each dataset and provide an overall picture of their performance.

For oversampling with replacement, the minority class examples are oversampled to the number at which the ratio of the number of two classes examples is 1. For SMOTE, the parameters are set as the following: number of nearest neighbors $K = 5$, percentage of oversampling $N = 200$. For ADASYN, the parameters are set as the following: number of nearest neighbors $K = 5$, desired balance level $\beta = 1$. The following tables contain the experiment results.

4.2.1 Ionosphere dataset

Naive Bayes						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	0.8826	0.8556	0.8180	0.8335	0.8656	0.9078
REP	0.8811	0.8489	0.8235	0.8330	0.8659	0.9002
SMOTE	0.8874	0.8481	**0.8479**	0.8450	0.8772	**0.9155**
ADASYN	**0.8964**	**0.8690**	0.8454	**0.8545**	**0.8833**	0.9134
Decision Tree						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	0.8726	**0.8450**	0.8002	0.8138	0.8517	0.8855
REP	**0.8745**	0.8441	0.8027	**0.8187**	**0.8551**	**0.8946**
SMOTE	0.8428	0.7460	**0.8566**	0.7948	0.8447	0.8847
ADASYN	0.8510	0.7669	0.8516	0.8022	0.8494	0.8725
Random Forest						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.9301**	**0.9276**	0.8721	**0.8981**	0.9157	0.9546
REP	0.9297	0.9230	0.8770	0.8983	0.9166	0.9572
SMOTE	0.9220	0.8745	**0.9140**	0.8925	**0.9201**	**0.9693**
ADASYN	0.9256	0.8971	0.8957	0.8953	0.9182	0.9650
SVM						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	0.9151	**0.9812**	0.7763	0.8661	0.8771	0.9273
REP	**0.928**	0.9674	0.8259	**0.8904**	**0.9013**	**0.9352**
SMOTE	0.9088	0.8887	**0.852**	0.8685	0.8948	0.9219
ADASYN	0.9169	0.931	0.8288	0.8762	0.8942	0.9261

Table 4.2: Ionosphere dataset results

4.2.2 Glass1 dataset

Naive Bayes						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.5864**	**0.4543**	0.7881	0.5648	0.5801	0.5873
REP	0.5753	0.4448	0.8202	0.5713	0.5901	0.6060
SMOTE	0.5637	0.4404	0.8540	0.5784	0.5836	**0.6354**
ADASYN	0.5697	0.4446	**0.8558**	**0.5829**	**0.5943**	0.6268
Decision Tree						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	0.6869	0.5711	0.5704	0.5562	0.6473	0.6727
REP	**0.7228**	**0.6106**	0.6284	**0.6116**	**0.6932**	**0.7071**
SMOTE	0.6566	0.5171	**0.7155**	0.5945	0.6647	0.6839
ADASYN	0.6774	0.5452	0.6347	0.5779	0.6595	0.6677
Random Forest						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.8041**	**0.7904**	0.6263	0.6914	0.7502	0.8102
REP	0.7968	0.7616	0.6424	0.6892	0.7510	0.8141
SMOTE	0.7493	0.6246	**0.7704**	0.6834	0.7526	0.8269
ADASYN	0.7890	0.7111	0.7049	**0.7012**	**0.7655**	**0.8361**
SVM						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	0.7073	**0.6905**	0.4065	0.4676	0.5688	0.7302
REP	**0.7166**	0.6107	0.6017	0.5959	**0.6809**	**0.7463**
SMOTE	0.5994	0.4668	**0.8620**	0.6022	0.6238	0.7426
ADASYN	0.6598	0.5203	0.7547	**0.6103**	0.6725	0.7235

Table 4.3: Glass1 dataset results

4.2.3 Pima India Diabetes dataset

Naive Bayes						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.7471**	**0.6560**	0.5868	0.6184	0.6987	0.7800
REP	0.7429	0.6330	0.6344	0.6329	0.7123	0.7848
SMOTE	0.7255	0.5849	**0.7520**	**0.6569**	**0.7307**	**0.7971**
ADASYN	0.7333	0.6050	0.6894	0.6437	0.7216	0.7833
Decision Tree						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.7342**	**0.6361**	0.5774	0.6015	0.6857	0.7617
REP	0.7267	0.6085	0.6294	0.6164	0.6991	0.7711
SMOTE	0.6975	0.5472	**0.8114**	**0.6523**	**0.7151**	**0.7789**
ADASYN	0.7092	0.5707	0.7106	0.6308	0.7081	0.7671
Random Forest						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.7530**	**0.6770**	0.5685	0.6165	0.6957	0.8123
REP	0.7517	0.6633	0.5953	0.6261	0.7050	**0.8159**
SMOTE	0.7430	0.6081	**0.7551**	**0.6725**	**0.7454**	0.8130
ADASYN	0.7478	0.6277	0.6917	0.6570	0.7331	0.8074
SVM						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.7564**	**0.7012**	0.5365	0.6061	0.6846	0.7812
REP	0.7420	0.6338	0.6276	0.6296	0.7097	**0.8082**
SMOTE	0.7178	0.5704	**0.8003**	**0.6647**	**0.7336**	0.8050
ADASYN	0.7304	0.5926	0.7417	0.6578	0.7324	0.8017

Table 4.4: PID dataset results

4.2.4 Vehicle dataset

Naive Bayes						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.6594**	0.3933	0.8749	0.5422	0.7206	0.7801
REP	0.6589	0.3963	0.9097	0.5516	0.7281	0.7806
SMOTE	0.6623	**0.4012**	**0.9364**	**0.5613**	**0.7365**	**0.7927**
ADASYN	0.6539	0.3874	0.8574	0.5332	0.7125	0.7645
Decision Tree						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	0.9179	**0.8307**	0.8136	0.8195	0.8779	0.9455
REP	**0.9217**	0.8052	0.8756	**0.8373**	0.9045	**0.9606**
SMOTE	0.9126	0.7778	0.8756	0.8216	0.8986	0.9492
ADASYN	0.9154	0.7763	**0.8953**	0.8297	**0.9078**	0.9449
Random Forest						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	0.9643	**0.9253**	0.9211	0.9225	0.9487	0.9875
REP	**0.9653**	0.9229	0.9290	**0.9252**	0.9522	0.9879
SMOTE	0.9605	0.8906	0.9472	0.9173	0.9558	**0.9893**
ADASYN	0.9600	0.8877	**0.9484**	0.9165	**0.9559**	0.9883
SVM						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	0.9630	**0.9217**	0.9191	0.9197	0.9472	0.9856
REP	**0.9695**	0.9022	0.9737	**0.9363**	**0.9709**	**0.9902**
SMOTE	0.9606	0.8687	0.9781	0.9198	0.9667	0.9889
ADASYN	0.9584	0.8583	**0.9831**	0.9162	0.9669	0.9887

Table 4.5: Vehicle dataset results

4.2.5 Vowel dataset

Naive Bayes						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.8595**	0.3245	0.5052	0.3926	0.6701	0.8637
REP	0.8185	0.3055	0.7883	0.4394	0.8037	**0.8777**
SMOTE	0.8413	**0.3279**	0.7116	**0.4474**	0.7786	0.8772
ADASYN	0.8069	0.2960	**0.8144**	0.4331	**0.8097**	0.8603
Decision Tree						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.9367**	**0.6726**	0.6254	0.6368	0.7730	0.8771
REP	0.9354	0.6240	0.7703	**0.6838**	0.8548	**0.9346**
SMOTE	0.9337	0.6284	0.7275	0.6652	0.8307	0.9065
ADASYN	0.9158	0.5317	**0.8037**	0.6354	**0.8620**	0.9151
Random Forest						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	0.9682	**0.9710**	0.6743	0.7912	0.8181	0.9824
REP	0.9749	0.9496	0.7661	0.8452	0.8721	0.9639
SMOTE	**0.9792**	0.9339	0.8313	**0.8773**	0.9080	**0.9879**
ADASYN	0.9767	0.8789	**0.8669**	0.8705	**0.9244**	0.9717
SVM						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	0.9428	**0.8453**	0.4660	0.5883	0.6729	0.9715
REP	**0.9685**	0.7586	0.9694	**0.8487**	**0.9686**	**0.9900**
SMOTE	0.9582	0.7195	0.9074	0.7969	0.9338	0.9820
ADASYN	0.9562	0.6840	**0.9762**	0.8022	0.9649	0.9847

Table 4.6: Vowel dataset results

4.2.6 Glass2 dataset

Naive Bayes						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	0.4696	0.1011	0.8075	0.1691	0.5940	0.6372
REP	0.4588	0.0998	**0.8159**	0.1772	0.5889	0.6508
SMOTE	**0.5117**	**0.1012**	0.7306	0.1766	0.5950	0.6473
ADASYN	0.5020	0.1011	0.7491	**0.1791**	**0.6060**	**0.6579**
Decision Tree						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.9107**	**0.2452**	0.2392	0.2220	0.4590	0.5300
REP	0.8721	0.2412	0.3620	0.2585	0.5521	0.6517
SMOTE	0.8449	0.2291	0.3865	0.2672	0.5664	0.6440
ADASYN	0.8417	0.2129	**0.4177**	**0.2758**	**0.5899**	**0.6639**
Random Forest						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.9252**	**0.7262**	0.1612	0.2480	0.3990	0.6795
REP	0.9151	0.4357	0.2037	0.2501	0.4312	**0.7536**
SMOTE	0.8922	0.2963	**0.3070**	**0.2756**	**0.5166**	0.7874
ADASYN	0.8777	0.2479	0.2995	0.2511	0.5068	0.7304
SVM						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.9245**	0.3012	0.0334	0.2388	0.5312	0.6587
REP	0.8768	**0.3079**	0.5414	**0.3817**	0.6811	**0.7329**
SMOTE	0.8465	0.2324	0.4342	0.2781	0.5913	0.7080
ADASYN	0.8210	0.2311	**0.6166**	0.3295	**0.7074**	0.7172

Table 4.7: Glass2 dataset results

4.2.7 Abalone dataset

Naive Bayes						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.8528**	**0.2016**	0.4826	0.2166	0.6466	0.7186
REP	0.7305	0.1280	0.6280	0.2109	0.6766	0.7242
SMOTE	0.7714	0.1413	0.5781	**0.2245**	0.6690	**0.7284**
ADASYN	0.6328	0.1071	**0.7387**	0.1861	**0.6776**	0.7202
Decision Tree						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.9337**	**0.3731**	0.1880	0.2681	0.4624	0.7545
REP	0.9071	0.2796	0.3623	0.3038	0.5765	0.7496
SMOTE	0.9117	0.2981	0.3649	**0.3168**	0.5793	0.7468
ADASYN	0.8499	0.1934	**0.4999**	0.2749	**0.6551**	**0.7554**
Random Forest						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	**0.9472**	**0.7198**	0.1366	0.2249	0.3619	0.7678
REP	0.9432	0.5431	0.1874	0.2680	0.4226	0.7768
SMOTE	0.9394	0.4790	0.2810	**0.3420**	0.5207	**0.7966**
ADASYN	0.9083	0.2862	**0.3873**	0.3228	**0.5990**	0.7946
SVM						
Methods	OA	Precision	Recall	F-measure	G-mean	AUC
NONE	0.9451	**0.8914**	0.0398	0.1241	0.2528	0.8327
REP	0.9153	0.3502	0.5368	**0.4172**	0.7063	0.8490
SMOTE	**0.9497**	0.6392	0.3153	0.4098	0.5530	**0.8555**
ADASYN	0.8744	0.2580	**0.6162**	0.3593	**0.7388**	0.8434

Table 4.8: Abalone dataset results

4.3 Discussion

Above datasets are sorted by the ratio of the number of minority class examples to the number of majority class examples from large to small. For each dataset, four classifiers and four oversampling methods are used so there are a total of 16 different imbalanced learners. For each learner, six assessment metrics are used, and the best performance is highlighted in bold. However, we should focus on the last three measurements, F-measure, G-mean and AUC, as discussed in Chapter 2. The first three evaluation metrics, overall accuracy, precision, and recall, are included in the tables because, as a reference, this can show that they are inappropriate to be used in imbalanced learning. Using classifiers without any oversampling method tends to have a higher overall accuracy rate and precision. This happens simply because there is no artificially data points in the training set, which can better detect the majority class examples. The training model is biased toward the majority class, thus it may cause high overall accuracy rate and precision. However, as mentioned in the previous chapters, these metrics are inappropriate for evaluating the performance.

From experiment results above, learning without using oversampling methods outperform under these traditional evaluation metrics; while they underperform under those regular imbalanced evaluation metrics. The results also show that in all datasets used in this paper, oversampling methods provides a better performance. This is a significant evidence to show the improvement of exercising these methods in imbalanced classification. There is no one oversampling method that absolutely outperforms the others. The performance of these sampling methods vary among datasets and classifiers. As the distribution of dataset became more imbalanced, oversampling methods have greater impact on the performance. For example, the last three datasets are considered more imbalanced relatively, and oversampling methods improve the performance significantly compared to first

four datasets, which are considered to be more "balanced".

The oversampling methods indeed improve the performance of imbalance learners in most of applications; however, the impact of choosing a suitable classifier is also important. The results indicate that there is a relatively huge variance among different classifier. For example in the abalone dataset as shown in Table 4.8, the areas under the curve(AUC) are about 0.72, 0.74, 0.78, and 0.84 for naive bayes, decision tree, random forest and svm, respectively. On the other hand, the the difference between the same metric under the same classifiers is relatively small, which is about $\pm 1\%$ variation. This phenomenon indicates that even though the oversampling methods improve the performance of classifiers, they can not effectively solve the imbalanced problems. Choosing the best combination of sampling methods and classifiers based on the distribution of minority class and majority class is essential to deal with imbalanced problems.

CHAPTER 5

Conclusion

This paper provides an overview of the usage of various classification algorithms combined with several oversampling methods for imbalanced dataset. Because of the natural problem of imbalanced learning, general classification methods will not provide good performance, and traditional evaluation metrics are inappropriate. Thus, classification of imbalanced dataset should be treated very differently. Several common assessment methods should be used in order for an imbalanced learner to give a comprehensive evaluation. Among those sampling techniques, oversampling is the most common one that reduces the bias. In most of the cases, classifiers with oversampling clearly perform better than those without. However, the impact of classifiers is relatively more sensitive than that of sampling techniques. One should select an appropriate classifier before determing which sampling method to use in order to improve the performance, which depends on the distribution of the data. Among those oversampling methods discussed above, there is no obvious evidence to show one is outperforming than the others. We hope this paper provide clear insights of the fundamental nature and solutions of the imbalanced learning problem, and can help the direction for future research of this field.

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(1):321–357, 2002.

[2] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119. Springer, 2003.

[3] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

[4] Vaishali Ganganwar. An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, 2(4):42–47, 2012.

[5] Thien M Ha and Horst Bunke. Off-line, handwritten numeral recognition by perturbation method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(5):535–539, 1997.

[6] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *Advances in intelligent computing*, pages 878–887. Springer, 2005.

[7] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. 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, 2008.

[8] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.

[9] Taeho Jo and Nathalie Japkowicz. Class imbalances versus small disjuncts. *ACM SIGKDD Explorations Newsletter*, 6(1):40–49, 2004.

[10] V García JS Sánchez RA Mollineda and R Alejo JM Sotoca. The class imbalance problem in pattern classification and learning. 2007.

[11] Hien M Nguyen, Eric W Cooper, and Katsuari Kamei. Borderline oversampling for imbalanced data classification. *International Journal of Knowledge Engineering and Soft Data Paradigms*, 3(1):4–21, 2011.

[12] Foster Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine learning*, 42(3):203–231, 2001.

[13] Yuchun Tang, Yan-Qing Zhang, Nitesh V Chawla, and Sven Krasser. Svms modeling for highly imbalanced classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(1):281–288, 2009.

[14] Juanjuan Wang, Mantao Xu, Hui Wang, and Jiwu Zhang. Classification of imbalanced data by using the smote algorithm and locally linear embedding. In *Signal Processing, 2006 8th International Conference on*, volume 3. IEEE, 2007.

[15] Mingrui Wu and Jieping Ye. A small sphere and large margin approach for novelty detection using training data with outliers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(11):2088–2092, 2009.