Complexity Measurement for Dealing with Class Imbalance Problems in Classification Modelling

A thesis submitted in fulfilment of the requirements

for the degree of Doctor of Philosophy

Massey University, 2012

Muhammad Nafees Anwar

Institute of Fundamental Sciences

Massey University

New Zealand

To my beloved late Mother

Abstract

The class imbalance problem is a challenge in the statistical, machine learning and data mining domains. Examples include fraud/intrusion detection, medical diagnosis/monitoring, bioinformatics, text categorization, insurance claims, and target marketing. The problem with imbalanced data sets is that the conventional classifiers (both statistical and machine learning algorithms) aim at maximizing overall accuracy, which is often achieved by allocating all, or almost all, cases to the majority class. Thus there tends to be bias against the minority class in class imbalance situations.

Despite numerous algorithms and re-sampling techniques proposed in the last few decades to tackle imbalanced classification problems, there is no consistent winning strategy for all data sets (neither in terms of sampling, nor learning algorithm). Special attention needs to be paid to the data in hand. In doing so, one should take into account several factors simultaneously: the imbalance rate, the data complexity, the algorithms and their associated parameters. As suggested in the literature, mining such datasets can only be improved by algorithms tailored to data characteristics; therefore it is important and necessary to do data exploratory analysis before deciding on a learning algorithm or re-sampling techniques.

In this study, we have developed a framework "Complexity Measurement" (CM) to explore the connection between the imbalanced data problem and data complexity. Our study shows that CM is an ideal candidate to be recognized as a "goodness criterion" for various classifiers, re-sampling and feature selection techniques in the class imbalance framework. We have used CM as a meta-learner to choose the classifier and under-sampling strategy that best fits the situation. We design a systematic over-sampling technique, Over-sampling using Complexity Measurement (OSCM) for dealing with class overlap. Using OSCM, we do not need to search for an optimal class distribution in order to get favorable accuracy for the minority class, since the amount of over-sampling is determined by the complexity; ideally using CM would detect fine structural differences (class-overlap and small disjunct) between different classes.

Existing feature selection techniques were never meant for class imbalanced data. We propose Feature Selection using Complexity Measurement (FSCM), which can specifically focus on the minority class, hence those features (and multivariate interactions between predictors) can be selected, which form a better model for the minority class.

Methods developed have been applied to real datasets. The results from imbalanced datasets show that CM, OSCM and FSCM are effective as a systematic way of correcting class imbalance/overlap and improving classifier performance. Highly predictive models were built; discriminating patterns were discovered, and automated optimization was proposed. The methodology proposed and knowledge discovered will benefit exploratory data analysis for imbalanced datasets. It may be taken as a judging criterion for new algorithms and re-sampling techniques. Moreover, new data sets may be evaluated using our CM criterion in order to build a sensible model.

Acknowledgements

I am highly in debt to Dr. Geoff Jones for his invaluable guidance and help to me in the development of my research skills, and his efforts to keep a stray bird on the right track.

I am very grateful to Dr. Siva Ganesh. Ganesh has always allowed me complete freedom to define and explore my own directions in research. While this proved difficult and somewhat bewildering to begin with, I have come to appreciate the wisdom of his way, as it encouraged me to think for myself.

I am thankful to Dr. Martin Hazelton and the Statistic Group who have provided me with financial support during my degree. They have kindly provided casual teaching assistantship positions and travel funds to attend conferences.

I thank Dr. Ganes Ganesalingam and Dr. Mark Bebbington for providing valuable feedback on parts of this thesis.

I also thank Mr. Peter Lewis for providing technical support.

Special thanks must also go to my family, especially to my late mother and to my wife Wa. They have provided unconditional support and encouragement through both the highs and lows of my time in Ph.D study.

Finally I would like to thank Higher Education Commission (HEC), Pakistan, for providing me with a scholarship to study at Massey University, New Zealand.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The class imbalance problem is a challenge in the statistical, machine learning and data mining domains, and significant research has been done in the last few decades to overcome the issues. A classifier suffering from class imbalance for a specific data set would see high accuracy overall but very poor performance on the minority class [1]. The problem can appear in two different types of data sets: two-class problems, where one class has many more instances than the other [2], and multi-class problems where each class contains a small fraction of the samples and we use one-vs-rest classification. Data meeting one of these two criteria may have different misclassification costs, either implicitly or explicitly for the different classes [3], and these kind of data sets exist in many "real-world" problems. Examples include fraud/intrusion detection [4], medical diagnosis/monitoring [5], bioinformatics [6], text categorization [7], insurance claims and target marketing [8]. The skewness of the class distribution can be severe. Some imbalanced data sets will only have 1% or less minority class examples. In the literature, the problem of imbalance is also known as dealing with a rare class or with skewed data [9].

Many statistical and machine learning algorithms have been found to produce unsatisfactory classification (low detection for the minority classes) for imbalanced data sets. They tend to over-fit the majority class and regard minority class examples as noise, of which the misclassification cost can be

extremely high or even fatal. The machine learning algorithms are poor to some extent because of the way they are designed, and to some extent because of the inappropriate performance measurements or evaluation metrics they use [9, 10]. Firstly, modern classifiers assume that unseen data points on which the classifier makes the prediction are drawn from the same distribution as the training data [11]. If testing and validation samples are drawn from different distributions, the model may give poor results because of a faulty model [12]. Although many classifier algorithms assume an even distribution in the whole data set, training and test sets may not necessarily have the same distribution [10]. Secondly, standard classifier algorithms are designed to maximize overall accuracy by minimizing the overall error rate. For example, logistic regression attempts to minimize the squared error rate and support vector machine (SVM) tries to minimize regularized hinge loss. Thirdly misclassification costs for different classes are different and may not be known during the modeling phase [9].

A classifier based on above mentioned assumptions will always produce poor accuracy (no or low detection for minority class) on an imbalanced data set. A classifier that attempts to classify minority class examples correctly will very likely see a significant decline in overall accuracy [11], because in a class imbalance scenario, the overall accuracy of the classifier is underrepresenting the value of classification accuracy of the minority class as compared to the majority class. In most cases, we would prefer a classifier that increases the accuracy of minority class even at the expense of majority class accuracy.

From the perspective of performance evaluation, overall accuracy might not be suitable, because class distribution and misclassification costs are rarely uniform [4] and use of such measures might be misleading. The evaluation criteria: accuracy and error rate: assume equal misclassification costs [4, 13] which is not true for imbalanced data. Evaluation matrices that take imbalance into account can improve classifier selection [10]. Alternative measurements, details of which are given later, include ROC analysis, AUC, precision, recall, geometric mean, F-measure, Precision Recall Break Even Point (PRBEP) [9] and Matthews Correlation Coefficient [14]. Researchers

use statistics like G-Mean [13] and area under receiving operating characteristics (AUROC) [11], to better evaluate minority class performance.

The severity of the problem depends on the degree of class imbalance, the overall size of the training data, and the model classifier involved. However, although many of the popular statistical or machine learning techniques have been tried in imbalanced situations, e.g. Classification Trees (CT) [15], Neural Networks (NN) [16], Support Vector Machines (SVM) [17], k-Nearest Neighbor (kNN) [18] and Random Forests [19], none of them has been found to be superior in general.

There have been many attempts to resolve the class imbalance problem. Apparently, the most common approach for resolving class imbalance problems is to use some form of re-sampling to adjust the imbalance in the training data [20]. Other approaches include: modification of existing algorithms, cost sensitive learning, one class learning or a combination of the above [21, 22].

Theoretically [23] and empirically, there is general understanding that different types of data require different kinds of classification. Not every imbalanced data set can be a problem for learning [24, 25]. It seems that the problem is rather with the small class in the presence of other factors such as class overlap [22]. The experiment conducted by [24] also suggested that degradation in classification modeling is not directly related to class imbalance alone, but rather to small disjunct (small subsets of one class that are separated from the largest subset of the same class).

1.1 Approaches for Dealing with Class Imbalance

This section provides an overview of the most prominent approaches for dealing with class imbalance problems. The first part of the discussion focuses on the limitations of traditional learning methods. It is followed by preprocessing strategies and algorithmic approaches.

1.1.1 Limitations of Traditional Learning Techniques

As shown in the literature [24, 26, 27] most traditional machine learning methods are affected by class imbalance. Among the existing methods, some are affected more seriously than the others. Classification trees appear to perform the worst when the data is imbalanced. This major drawback is thought to be rooted in the pruning stage [28]. Conclusions found in Visa and Ralescu [9] suggest that support vector machines (SVMs) algorithm is not affected by the imbalance problem. Another study by Japkowicz [11] found that the artificial neural networks (ANN) and SVMs are not sensitive to the class imbalance problem, while decision trees are strongly affected by the imbalance. These observations suggest the need for a goal of creating a classifier that performs well over a range of situations [29]. The alternative solution is to alter the data distribution (via preprocessing) such as to provide a more appropriate class distribution in the training set.

1.1.2 Pre-processing Techniques

At the data level, the objective is to re-balance the class distribution by re-sampling the data. This is the most direct solution for dealing with an imbalanced problem; i.e., to class-balance the training data. Numerous algorithms have been developed, with the majority of efforts focused on proposing and evaluating sampling techniques comparatively. However, it is important to mention that many of them are effective only in certain situations (i.e. type of rarity), and for a specific classifier. Here we describe the most important sampling strategies.

Sampling techniques follow two major strategies: over-sampling and under-sampling. Over-sampling reduces the imbalance rate by duplicating examples from the under-represented class for inclusion in the training set. Important drawbacks of over-sampling refer to the fact that it may lead to over-fitting, increase the time required to build the classifier, and give no information gain from the learning process [31]. Under-sampling performs balancing by removing examples from the majority class. While this is effective in correcting class imbalance by reducing the data size, it may cause loss of information.

An informative under-sampling scheme (not random but selected to serve some purpose) could overcome this drawback. There is general disagreement among researchers regarding which technique is better, some researchers being in favor of over-sampling [2] while others advocating under-sampling [32].

However, sampling should be considered in the broad scenario consisting of the problem itself and an appropriate classifier for the problem. Different classifiers perform better in conjunction with specific sampling techniques [27]. Hulse et al. [27] stated that random under sampling (RUS) worked well for classification tree (CT), while random over-sampling (ROS) being more suitable for logistic regression (LR). Another important aspect which influences sampling is the noise associated with the data, which has been shown to affect the minority class more so than the majority class [10].

A comprehensive study of sampling techniques can be found in [26, 27]. Several informative over-sampling and under-sampling techniques are compared. Sophisticated methods, resulting from the combination of different basic techniques are briefly described, and thoroughly evaluated on benchmark data sets [33]. Although most of them yielded an increase in Classification accuracy of minority class, as expected, there was no technique which dominated all the others on all data sets. Moreover, based on the results of the evaluations, it was concluded that "random over-sampling provided competitive results with the more complex methods". Another important study by Japkowicz and Stephen [2] suggests that the improvements produced by under/over sampling are more significant for highly imbalanced datasets.

A very important issue regarding sampling techniques refers to the appropriate amount of over/under sampling required [35]. A best class distribution is usually unknown and needs to be investigated [21, 34]. An appropriate amount of over/under sampling is a non trivial problem, whether it is domain (learning algorithm) and/or data-dependent. Most classification algorithms adapt the class distribution of the available data. This assumption could be wrong for the following reason: the real class distribution could be unknown and it may not be the most appropriate class distribution for learning [36]. Hall and Joshi [35] suggested that using wrappers i.e., performing a search for the correct percentage of under-sampling of the majority class or synthetic

over-sampling of the minority class, may improve the performance of a particular classifier. Since this is a wrapper technique, the correct proportion of under-sampling or over sampling may not be suitable for another classifier. Another study by Weiss and Provost [36] proved that if there is a best distribution for the training set, it needs more examples from the minority class to form an appropriate model from the training set. Moreover, the same performance could be acquired with a smaller training set, if the distribution is balanced in favor of the minority class. It has been reported that over/under sampling solved the 'imbalance problem' in some studies, but it did not help in other studies [13, 37]. Possible reasons are that an inappropriate class distribution for the data set, or other data features yet to be explored, has degraded classifier performance.

Another data-related mechanism in handling imbalance is repetitive re-sampling techniques designed to alleviate some of the problems associated with re-sampling [38]. Repetitive re-sampling constructs an ensemble of models, each using a different sample from the majority class, hence reducing the loss of information when only one subset of the majority class is used. Cluster-based re-sampling techniques can be used in order to select majority class examples from different regions of the data set. This way, we may isolate the minority class in a single partition, which is assigned special treatment, while for the other(s) regular reasoning methods (such as data segmenting of majority class) are employed [10].

Re-sampling can be done in more sophisticated ways: Synthetic Minority Over-sampling Technique (SMOTE) [31] creates and interpolates new minority class examples; Cluster-based over-sampling [24] rectifies class imbalances between the majority class clusters and the minority class; Condensed nearest neighbor (CNN) [39] chooses data near to the decision boundary to build a sensible model for classification; Adaptive sampling is phase-wise learning, where at each iteration a number of examples from the data set is added to the model. In these techniques all those examples which have been misclassified by a certain classifier are added to the final model [40]. Progressive sampling [41] is a method for under-sampling of majority class by progressively increasing the majority class sample size for training the model, as

long as model accuracy improves. Importance sampling [42] concentrates on the examples near to the decision boundary. The difference between the CNN and Importance sampling is that CNN is based on nearest neighbor technique and Importance sampling is a wrapper technique, i.e., it select all those example which are misclassified by a certain classifier.

Cost-sensitive learning has been shown to outperform any kind of sampling in class imbalance problems by increasing the accuracy of minority class [43]. However, if the costs are not known or are not constant, it is not applicable [44]. The importance of cost sensitive learning is related to the cost of acquiring data. Weiss et al. [43] demonstrated that for large data sets a cost-sensitive approach did consistently better on minority class than any sampling technique, but performed poorly for small data sets, for which the cost information could not be accurately estimated. Unless the misclassification cost is implemented to adapt to local density or small disjunct, cost sensitive learning will not improve the classifier's performance.

One-class learning has been claimed to be advantageous for imbalanced data sets as it avoids over-fitting on the majority class [45]. One-class SVM [46], SHRINK [1], and RIPPER [5] are examples of one class learning. A good method could be to generate a model for each class, rather than generating a complete model. However experiments from various researchers [18, 27] proved that a classifier trained using only minority class examples will not be as good as the result using minority and majority class examples. Thus, unless one has only training samples known to be from one class and no other information, one-class learning is unlikely to be a better approach.

Among the various pre-processing tasks, feature selection is the newest of the techniques attempting to resolving the class imbalance problem. It not only reduces the data dimensionality, by removing variables, thus reducing the search space, but it also improves the classification models most of the time [7]. Feature selection could be beneficial in such cases, by selecting the features which can capture the high skew in the class distribution [9, 44]. Most of the research to date in feature selection for the class imbalance scenario has been focused on text categorization [7, 48, 49]. One aim of this thesis is to investigate feature selection in a wider range of applications, using

various feature selection techniques.

1.2 Complexity Measurement Approach

Researchers have developed many techniques to attempt to overcome the class imbalance problem, including re-sampling, new algorithms and feature selection. Researchers working on different methods in different domains have yet to agree on standard benchmark data sets or on a systematic approach to solving the problem. Given the suggestion that the skewed class distribution may not be the only problem [10, 22, 24, 25], we see the need for a systematic way of investigating and explaining what intrinsic features of the data are affecting the degraded learning performance of an imbalanced data set. We think that the answer could be found through exploratory data analysis (EDA) or data complexity analysis, and that data complexity measures can be used to devise a structured study for learning about class imbalance problems. To our knowledge, no researcher has used complexity of a data set as a criterion to devise a systematic strategy to overcome the class imbalance problem. Some authors have considered complexity measurement [50, 51, 52, 53] but never as a way to cater for imbalance, rather to indicate the overall level of difficulty in the data sets.

The two workshops [29, 30] on Imbalanced Data Sets focused on three aspects: sampling, new algorithm and feature selection; this thesis makes new contributions associated with each of these areas, as outlined below.

1.2.1 Overview of Literature Review (Chapter 2)

This chapter reviews in detail the existing approaches to combating the class imbalance problem. The references cited in the chapter will cover the major theoretical issues, from which we identify some areas yet to be explored.

1.2.2 Overview of Complexity Measurement (Chapter 3)

It has been reported that re-sampling solved the imbalanced 'problem' in some studies, but it did not help in other studies [13, 37]. Jo and Japkowicz [24] suggested that class imbalance may yield small disjunct, causing degradation in classification accuracy of minority class. Similarly, Prati et al. [54] argued that class imbalance in the presence of class overlap causes problem. We thought that Complexity Measurement (CM) would be a good approach to clarifying what features of a data set lead to degradation of classification.

Data complexity can be described as an indicator that shows the level of difficulty in class learning for a specific data set. Ho and Basu [50] proposed some complexity measures for two-class classification problems. They defined complexity measurement under three different headings: overlaps of feature spaces from different classes, separability of classes and measures of geometry, topology and density of manifolds. This study has been widely used subsequently. In the literature we can see a number of studies using data complexity as a measure to define the strength of their methodology or to make a comparison between different classifiers using one or more of the complexity measures introduce by Ho and Basu [50]. But their data complexity measurement was never meant for the class imbalance problem, and we found it an inadequate measurement for this situation, which is shown in Chapter 3. This implied a good reason to introduce a new Complexity Measurement (CM) for class imbalance problems and utilize CM to tackle class imbalance problems in a systematic way.

1.2.3 Overview of Under-sampling Techniques in relation to Complexity Measurement (Chapter 4)

An important theoretical result related to the nature of class imbalance is presented in [2]. They concluded that the imbalance problem is a relative problem, which depends on: (1) the imbalance ratio, i.e. the ratio of the num-

ber of majority to the minority examples, (2) the complexity of the concept represented by the data, (3) the overall size of the training set and (4) the classifier involved. The experiments were conducted on artificially generated data, in an attempt to simulate different imbalance ratios, complexities and data set sizes. Hulse et al. [27] suggests that the usefulness of each particular re-sampling technique depends on various factors as mentioned by Japkowicz and Stephen [2] on real datasets. Several papers like [11] , [18], [21], [37] and [57], have also studied this dependence during the last decade.

In this chapter we perform an investigation using our novel approach Complexity Measurement (CM) on real world benchmark data sets, to study the effect of the class imbalance problem on several classes of algorithms: Decision Trees, Neural Network, Logistic regression, Support Vector Machines and ensemble methods. Our analysis focuses on the factors described in [2]: data complexity and learning algorithm: in an attempt to address some of the open questions presented there, related to the applicability of the conclusions drawn on artificial data in real-world settings. We conducted our experiments by evaluating various performance metrics (see Chapter 2). We applied various under-sampling techniques for improving the behavior of classifiers in imbalanced problems available in literature under different complexity groups. The results of this study (first of its kind) suggest that a more systematic analysis of classifier and re-sampling technique can be performed by considering data complexity.

1.2.4 Overview of Over-sampling Techniques using Complexity Measurement (Chapter 5)

As mentioned earlier, one of the recommendations of the Imbalanced Data Sets workshops was the need for a new classifier, which would work entirely on its own. The underlying difficulty associated with highly imbalanced classification problems suggests that an automatic classification system, working entirely on its own, may not be a realistic goal in the near future. Rather, we feel that a two step system will work better, with an automatic classification system to measure complexity (pre screening tool) followed by the

re-sampling procedure. The first (pre-screening) step of classification is to measure the complexity for the minority class. The aim is to make the execution of the more expensive re-sampling procedure (that is the second step) affordable as the pre-screening step effectively narrows down the minority class examples that need over-sampling. The specific goal of this chapter is to show how our complexity measurement can be used in the pre-screening step. Our study proves that it works better than the existing over-sampling techniques and automatically optimizes the class distribution without the need for a wrapper. This technique may be the answer to a very important issue regarding sampling techniques, i.e., the appropriate amount of over/under sampling required [21, 34, 35].

1.2.5 Overview of Feature Selection using Complexity Measurement (Chapter 6)

In this chapter we will review the existing techniques of feature selection and their application in the imbalance scenario. Researcher [7, 48, 49, 55] shows that in high dimensional data sets, feature selection can by itself combat the class imbalance problem. Our aim is to discover which feature selection technique will be the most effective in increasing the classification accuracy of the minority class. We avoid here the extreme feature selection problems, such as microarray analysis [56] and text classification [48] which have more features than cases. Our focus is on the performance of the techniques in selecting from a relatively modest set of possible features when the main concern is the classification accuracy of the minority class.

In our comparative study we found that existing feature selection techniques did not produce satisfactory results on imbalanced data sets. We propose a novel feature selection algorithm using Complexity Measurement (CM) to evaluate the extracted low-level features. Using the best feature subset captured by Feature Selection using Complexity Measurement (FSCM), we compare the performance of the proposed framework with the performance of several other existing feature selection algorithms discussed above. We used benchmark data sets from UCI [33] in conjunction with the well-known

Classification Tree [15] classifier. Overall, our proposed framework performs better than other feature selection methods over the chosen classifier, and performs significantly better with highly imbalanced data sets.

1.3 Contribution of the Thesis:

When dealing with imbalanced classification problems there is no consistently superior approach for all data sets (in terms of re-sampling, algorithmic or feature selection techniques) in the existing literature. We believe Complexity Measurement (CM) could be regarded as an emerging field in the class imbalance research which pays attention to the distinctiveness of the particular data. We focus on a wider context by taking into account various factors, such as data complexity and the algorithms and their associated parameters. This thesis introduces a new complexity measure for imbalanced data and shows that, in classification modeling with class imbalanced problems, CM can play an important role and can be used to choose the appropriate classifier, feature selection and re-sampling technique that best fits the situation.

This thesis makes new contributions associated with each of these areas: sampling, new algorithm and feature selection.

The method, developed here are illustrated and tested by applying them to the UCI datasets [33], which are regarded as benchmark datasets in the literature. In our final chapter we briefly examine the feasibility of the proposed methodologies on much larger datasets.

References

[1] Liu, J., Hu, Q., Yu, D., A comparative study on rough set based class imbalance learning, Knowledge-Based Systems, Vol.21, no.8, pp. 753-763, 2008.

[2] Japkowicz N. and Stephen, S., The Class Imbalance Problem: A Systematic Study , Intelligent Data Analysis, Vol 6,no. 5,pp. 429-450, 2002.

[3] Elkan, C., The foundation of cost sensitive learing, Proceeding 17th International joint conference for Artificial Intelligence, pp. 973-978, 2001.

[4] Fawcett,T.and Provost, F., Adaptive Fraud Detection, Data Mining and Knowledge Discovery, Vol.1, no.3, pp. 291-316, 1997.

[5] Cohen, G., Hilario, M., Sax, H., Hugonnet, S., and Geissbuhler, A., Learning from imbalanced data in surveillance of nosocomial infection, Artificial Intelligence in Medicine, Vol. 37, no. 1, pp. 7-18, 2006.

[6] Garca-Pedrajas, N., Prez-Rodrguez, J., Garca-Pedrajas, M., Ortiz-Boyer, D., Fyfe, C., Class imbalance methods for translation initiation site recognition in DNA sequences, Knowledge-Based Systems, Vol. 25, no. 1, pp. 22-34, 2012.

[7] Zheng, Z., Wu, X., and Srihari, R., Feature Selection for Text Categorization on Imbalanced Data, ACM SIGKDD Explorations Newletter, vol 6, pp. 80-89, 1999.

[8] Zadrony, B. and Elkan, C., Learning and making decisions when costs and probabilities are both unknown. Proceeding of 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 204-214, 2001.

[9] Visa, S. and Ralescu, A., Issues in mining imbalanced data sets a review paper, Proceeding of Midwest Artificial Intelligence and Cognitive Science Conference (MAICS'05), pp. 67-73, 2005.

[10] Weiss, G.M., Mining with Rarity: A Unifying Framework. SIGKDD Explorations, Vol. 6, no. 1, pp.7-19, 2004.

[11] Japkowicz, N., Learning from Imbalanced Data Sets, Proceeding of AAAI'00, 2000.

[12] Forman, G. and Cohen, I., Learning from Little: Comparison of Classifier Given Little Training, Proceeding Eight European Conference Principle and Practice of knowledge Discovery in Databases, pp. 161-172, 2004.

[13] Kubat, M., and Matwin, S. Addressing the Curse of Imbalanced Data Sets: One Sided Sampling. In the Proceedings of the Fourteenth International Conference on Machine Learning, Nashville, pp. 179- 186,1997.

[14] Matthews, B.W., Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochimica et biophysica acta 405, no. 2 , pp. 442-451, 1975.

[15] Chawla, N., C4.5 and Imbalanced Data sets: Investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In Workshop on Learning from Imbalanced Datasets (ICML03), 2003.

[16] Nguyen Hieu T. and Smeulders, A., Active Learning Using Preclustering, In Proceeding of the 21^{st} International Conference on Machine Learning, Banff, Canada, pp. 79-86, 2004.

[17] Akbani, R., Kwek, S., and Japkowicz, N., Applying support vector machines to imbalanced datasets. In Lecture Notes in Computer Science, 3201, pp. 39-50, 2004.

[18] Zhang, J. and Mani, I., kNN Approach to Unbalanced Data Distributions: A Case Study involving Information Extraction. In Workshop on Learning from Imbalanced Datasets (ICML03), 2003.

[19] Chen, C., Liaw, A. and Breiman, L., Using random forest to learn imbalanced data. Technical Report Technical report 666, Statistics Department, University of California at Berkeley, 2004.

[20] Batista, G. E. A. P. A.,Prati, R. C., and Monard, M. C., A study of the behavior of several methods of balancing machine learning training data. SIGKDD Exploration, Vol. 6, no. 1, pp. 20-29, 2004.

[21] Estabrooks, A., Jo, T., and Japkowicz, N., A Multitple Resampling Method For Learning From Imabalanced Data Sets. Computational Intelligence, Vol. 20, no. 1, pp. 18-36, 2004.

[22] Prati, R. C., Batista, G. E. A. P. A., and Monard, M. C., Class imbalances versus class overlapping: an analysis of learning system behavior. In MICAI, pp. 312-321, 2004.

[23] Devroye L., Any discrimination rule can have an arbitrary bad probability of error for finite sample size. IEEE Trans Pattern Analysis and Machine Intelligence, Vol. 4, no. 2, pp. 154-157, 1982.

[24] Jo, T., Japkowicz, N., Class imbalances versus small disjuncts, SIGKDD Explorations, Vol.6, pp. 429-450, 2004.

[25] Provost, F., and Fawcett, T., Robust classification for imprecise environments, Machine Learning Journal, Vol. 42, no. 3, pp. 203-231, 2001.

[26] Batista,G.E., Prati, R.C., and Monard,M.C., Balancing strategies and class overlapping. In proceeding of 6th International Symposium on Intelligent Data Analysis, Madrid, Spain, pp.24-35, 2005.

[27] Hulse, J.V., Khoshgoftaar, T.M., Napolitano, A., Experimental perspectives on learning from imbalanced data, in: Proceedings of the 24th International Conference on Machine Learning, Corvalis, Oregon, pp. 935-942, 2007.

[28] Drummond, C. and Holte, R., C4.5, Class Imbalance, and Cost Sensitivity: Why Under-sampling beats Over-sampling. Workshop on Learning from Imbalanced Datasets II, ICML, Washington DC. 2003.

[29] Proceeding of AAAI'00, Workshop Learning from Imbalanced Data sets, Japkowicz, N., ed., 2000.

[30] ICML '03, Workshop Learning from Imbalanced Data sets, Chawal, N., Japkowicz, N. and Kolcz, A., 2003.

[31] Chawla, N., Bowyer, K.,Hall, K., Kegelmeyer, P., SMOTE: Synthetic Minority Over-Sampling Technique, Journal of Artificial Intelligence Research, Vol. 16, pp. 321-357, 2002.

[32] Garcia, S., Herrera, F., Evolutionary Undersampling for Classification with Imbalanced Datasets: Proposals and Taxonomy, Evolutionary Computation, Vol. 17, no. 3, pp. 275-306, 2009.

[33] UCI KDD archive.http://kdd.ics.uci.edu/databases/covertype/covertype.html. 2005.

[34] Chan, P.K. and Stolfo, S.J., Towards Scalable learning with non-uniform class and cost distribution. A case study in credit card fraud detection, in proceeding of the foruth Conference on Knowledge Discovery and Data Mining, New York, pp. 164-168, 1998.

[35] Hall, L. O., Joshi, A., Building Accurate Classifiers from Imbalanced Data Sets, IMACS05, Paris (2005).

[36] Weiss, G. and Provost, F., Learning when training data are costly: The effect of class distribution on tree induction. Joural of Artificial Intelligence Research, pp.315-354, 2003.

[37] Drummond, C. and Holte, R., Severe Class Imbalance: Why Better Algorithms Arent the Answer. In Proceedings of the 16th European Conference on Machine Learning (ECML/PKDD'05), pp. 539-546, 2005.

[38] Hulse, J.V., Khoshgoftaar, T.M., and Napolitano, A., An Empirical Comparison of Repetitive Undersampling Techniques, International Conference on Information Reuse and Integration, IRI '09. IEEE, pp. 29-34, 2009.

[39] Hart, P. E., The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory IT-14, pp. 515-516, 1968.

[40] Iyengar, V. S., Apte, C., AND Zhang, T. Active learning using adaptive resampling. In KDD 00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (New York, NY, USA), ACM Press, pp. 9198, 2000.

[41] Provost, F., Jensen, D., and Oates, T. Efficint Progressive Sampling. Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, San Diego, California, United States, pp. 23-32, 1999.

[42] Breiman, L., Pasting small votes for classification in large databases and on-line, Machine Learning, Vol. 36, pp.85-103, 1999.

[43] Weiss, G., McCarthy, K. and Zabar, B. Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs? In Proceedings of the 2007 International Conference on Data Mining, pp.35-41, 2007.

[44] Chawla N., Data mining for imbalanced datasets: an overview. In The Data mining and knowledge discovery handbook, Springer 2005.

[45] Vidrighin Bratu, C., Muresan T., Potolea, R., Improving Classification Accuracy through Feature Selection, In Proceedings of the 4th IEEE International Conference on Intelligent Computer Communication and Processing, ICCP, pp. 25-32, 2008.

[46] Manevitz, L. M. and Yousef, M., One SVMs for document classification, Journal of Machine Learning Research, Vol. 2, pp.139-154, 2001.

[47] Kubat, M. Holte, R. and Matwin, S., Machine Learning for the Detection of Oils Spills in Satellite Radar Images, Machine Learning, Vol 30, pp. 195-219, 1998.

[48] Forman, G.and Cohen, I., Learning from Litte: Comparison of Classifiers Given Little Training, Proceeding 8th European conference, principles and practice of knowledge Discovery in Database, pp.161-172, 2004.

[49] Mladenic, D., and Grobelnik, M., Feature Selection for Unbalanced Class distribution and Naive Bayes, Proceeding of 16th international conference on Machine learning, pp. 258-267, 1999.

[50] Ho, T. K., and Basu. M, Complexity measures of a supervised classification problems, IEEE Trans Pattern Analysis, Machine learning, Vol. 24, no. 3, pp. 289-300, 2002.

[51] Weng, C. G. and Poon, J., A data complexity anylysis on imbalanced datasets and an alternative imbalance recovering strategy, Proceeding of the IEEE/WIC/ACM International Conference on Web Intelligent, 2006.

[52] Li, Y., Member, S., Dong, M., Kothari, R., Classificability-based omnivariate decision trees, IEEE Trans Neural Netwrok, Vol. 16, no. 6, pp. 1547-1560, 2005.

[53] Bernado-Mansilla, E., Ho, T., K., Domain of competence of XCS classifier system in complexity measurement space. IEEE Trans Evol Computation, Vol. 9, no. 1, 82-104, 2005.

[54] Prati, R. C., Batisita, G. E., and Monard, M. C., Class Imbalances versus Class Overalpping, An Analysis of the Learning System Behaviour, in MICAI, pp. 312-321, 2004.

[55] Chen, J.J., Tsai, C.A., Young, J.F. and Kodell, R. L., Classification ensembles for unbalanced class size in predictive toxicology. SAR and QSAR in Envoirmental Research, Vol.16, no. 6, pp. 517-529, 2005.

[56] Xiong, H. and Chen, X., Kernel Based Distance Metric Learning for Microarray Data Classification, BMC Bioinformatics, Vol 7, no. 299, pp. 1-11, 2006.

[57] Garcia, V., Mollineda, R., Sanchez, J., S., On the k-NN performance in challenging scenario of imbalance and overlapping. Pattern Analysis Application Vol. 11, no. 3-4, pp. 269-280, 2008.

Chapter 2

Literature Review of Class Imbalance Problem

In two-group classification problems, class imbalance occurs when observations belonging to one class/group heavily outnumber the cases in the other class. In real-world applications, it has been often observed that class imbalance (significant differences in class prior probabilities) may produce severe deterioration of the classifier performance, in particular, with patterns belonging to the minority classes. In many such cases, the minority class is the class of interest. The severity of the deterioration depends on the degree of class imbalance, the concept and overall size of the training data, and the model classifier involved. Two main approaches have been suggested in the literature to tackle class imbalance problems. The first is to 'balance' the 'imbalance' in the class distribution via sampling schemes, especially over-sampling of minority cases and/or under-sampling the majority class. The second approach uses a cost-sensitive learning, mainly with high cost for misclassifying minority cases compared to majority cases. This chapter will review various techniques that have been proposed in the literature for handling class imbalance problems in classification modeling. The references cited in the chapter will cover the major theoretical issues, from which we identify some areas yet to be explored.

2.1 Introduction

Classification is known as discrimination is the statistical literature and supervised learning in the machine learning and data mining literature. Classification modeling is, to build a function/rule from the training data, and to use the rule to classify new data (with unknown class), into one of the existing categorical classes or groups. Classification is an important task both in Statistics and Data Mining. Various classification techniques, such as, classical discriminant functions, classification tree, neural network, Bayesian classifier/network, nearest neighbor, support vector machines, and various ensemble ideas, (e.g. bagging, boosting, random forests etc), have been well developed and successfully applied to many domains. Maximizing classification accuracy is the usual goal of these algorithms.

However, maximizing overall accuracy is not always the case in real world data where one class might be represented by a large number of examples, while the other is represented by only a few, such as oil spill detection [1], fraud detection [2], network intrusion detection [3]. Most classification techniques try to minimize the total misclassification error rate. They ignore the differences between types of misclassification errors. In particular, they assume that all misclassification errors cost equally. The result of these assumptions is that classification modeling on imbalanced data sets produces unsatisfactory classifiers. In many classifications tasks, it is the rare classes/cases that are of major interest (e.g., 'claimants' in Insurance data or industry). Learning algorithms that do not consider class-imbalance tend to be overwhelmed by the majority class and ignore the minority class [4]. Its importance grew as more researchers [2, 5, 6, 7, 8, 9, 10, 11] realized that imbalanced cases cause poor classification performance on the minority class, and that most algorithms behave badly when data sets are highly imbalanced.

Class imbalance problems can be found in many application domains, such as medical diagnosis (e.g., rare disease and rare genes mutations), networking monitoring and behaviour profiling -intrusion, fraud detection (credit card, phone call, insurance), risk management, helicopter gear-box fault mon-

itoring, shuttle system failure, earthquakes and nuclear explosions, text classification and oil spill detection [12]. From a practical application point of view, class imbalance can be categorized as 'natural imbalance', such as rare disease and fraud cases or as 'artificial imbalance', where one class is too expensive to obtain, such as shuttle system failure.

There has been a lot of research on the class imbalance problem in the last few decades. There have also been a few good review papers, such as Visa and Ralescu [12] who reviewed the current progress in the class imbalance field and argued that the poor performance of existing classification (machine learning) algorithms were due to three factors: using overall accuracy as the criterion, assuming an even class-distribution and using equal cost of misclassification error. Their reason given for this poor performance was that the algorithms used rarely suited real world applications. Kotsiantis et al. [13] reviewed various techniques for handling class imbalance problems, while Batista et al., [10] discussed several issues, such as class overlapping and various balancing strategies related to class imbalance problems. Weiss [14] paid attention to differences and similarities between the problems of rare classes (larger portion of rare cases) and rare cases (small subsets of one class that are separated from the largest subset of the same class, also known in literature as small disjunct [5]), and then discussed some of the common issues and solutions in modeling imbalanced data sets.

The structure of this chapter is organized as follows: Section 2.2 gives a review of two international conferences and a special issue of SIGKDD Explorations 2004, on class imbalance problems; Section 2.3 describes the various sampling techniques used in the literature to tackle the class imbalance problem; Section 2.4 describes evaluation measures associated with class imbalance and Section 2.5 gives overall conclusions and discusses opportunities for further research. Moreover, in the literature, researcher have used various classification algorithm such as: Logistic Regression, Classification Tree (also known as C4.5 algorithm), Neural Network, and Support Vector Machine (SVM) for their studies. For a description of how a particular algorithm works please see Appendix 2.7.

2.2 Importance of Class Imbalance Problems:

In recognition of class imbalance problems in various application domains, there were two conferences in Artificial Intelligence and one special issue in SIGKDD Exploration (2004) on dealing with the class imbalance problems. The first was held at the American Association for Artificial Intelligence (AAAI) conference in 2000 [15], and the second was held at International Conference on Machine Learning conference in 2003 (ICML'03)[16]. Weiss [14] reviewed these conferences and different approaches to tackle class imbalance problem in SIGKDD Explorations (2004).

At the AAAI 2000 workshop, the main debated topics from the workshop were: how to evaluate learning algorithms; what learning evaluation measures should be used; one class learning vs discriminant methods; various re-sampling methods; relation between class imbalance problems and cost sensitive learning; and the goal of creating a classifier that performs well over a range of situations.

The second workshops for the class imbalance problem was a part of ICML'03, where most of the research was based on the learning from the first workshop, such as using ROC or cost curves as the evaluation matrices rather than overall accuracy. Despite agreeing in the first workshops to look for a new classifier, most of the papers focused on tuning the parameters of the classification tree in order to perform better on imbalanced data sets, and various re-sampling techniques under various aspects contributed almost half of the papers and most debated topics. Japkowicz [6] questioned the fact that the within-class imbalance is responsible for the problem. The main idea of lack of representation of some important aspects (points near to decision boundary) of the minority class can be referred to as class overlap.

In summary, the learning from the two imbalance data sets workshops explored the approaches to overcome the class imbalance problem namely: sampling, feature selection, and new algorithms (new classifier).

The sixth issue of SIGKDD Exploration (2004) was dedicated to the class imbalance problem, in which Weiss [14] gave a review of research on class

imbalance techniques, while other papers in the volume dealt with the issues of re-sampling, feature selection, and one-class learning.

2.3 Dealing with Class Imbalance Problems

Four main approaches have been suggested in the literature to tackle class imbalance problems. The first is to 'balance' the 'imbalance' in the class distribution via sampling schemes, especially over-sampling of minority cases and/or under-sampling the majority class. The second approach uses a cost-sensitive learning, mainly with higher cost for misclassifying minority cases compared to majority cases. The justification for this is that the problem of imbalanced data is often associated with asymmetric costs of misclassifying elements of different classes. The third and fourth approaches involve ensemble models for final classification and the relatively new approach of feature selection. Class imbalance involving two classes has been the main focus of studies in the literature. Reported work focuses on three aspects of the class imbalance problem:

1. What is the nature of the class imbalance problem, i.e. in what situations can class imbalance hinder the performance of standard classifiers?

2. What are the possible solutions in dealing with class imbalance problems?

3. What are the proper evaluation measures of classification performance of the classifier?

2.3.1 Re-sampling Techniques

At the data level, the objective is to re-balance the class distribution by re-sampling the data. This is the most direct method ways for dealing with class distributions towards a more balanced distribution. These solutions include many different forms of re-sampling such as random over-sampling, random

under-sampling, active-sampling techniques (where random is replaced by informed re-sampling techniques) and combinations of the afore-mentioned techniques. Hulse et al. [17] suggest that the usefulness of each particular re-sampling technique depends on various factors, including the ratio between positive and negative examples, the characteristics of data, and the nature of the classifier. Several papers [4], [6], [18], [19] and [20] have also studied the approach of changing the class distributions. Japkowicz [21] compared various re-sampling strategies and concluded that under-sampling and over-sampling are very effective in dealing with the class imbalance problem.

2.3.1.1 Under-Sampling Techniques

The most simple method of under-sampling is Random Under-sampling [10]. Random Under-Sampling (RUS) is a non-heuristic method aimed to balance the data set by eliminating examples of the majority class. One of the disadvantages of RUS is that it can throw away potentially useful information, that could be useful for classifier.

There are many empirical under-sampling methods proposed, which can be named as Neighborhood Cleansing Techniques. These are based on the noise model hypothesis, which considers the examples near to the decision boundary of the two classes as noise to be eliminated.

Condensed Nearest Neighbor Rule Hart's Condensed Nearest Neighbor Rule (CNN) [22] is used to find a consistent subset of examples. It is based on the idea of a consistent subset of a sample set, which is a subset which can correctly classifies all of the remaining examples in the training sets, when used as a store reference to nearest neighbor rule. A subset $\widehat{X} \subseteq X$ is consistent with X if using a 1-Nearest Neighbor classifier (1-NN, i.e., minimum distance between data points, and if a case has a minimum distance to an example from a different class, this case will be missclassified), $\widehat{X}$ correctly classifies the examples in X. An algorithm to create a subset $\widehat{X}$ from X as an under-sampling method is define as follows: first, randomly draw one majority class example and all examples from the minority class and put these examples in $\widehat{X}$. Afterwards, use a 1-NN classifier over the examples in $\widehat{X}$

to classify the examples in X. Every misclassified example from X is moved to $\widehat{X}$. The notion behind this technique is to find a subset of the training set, which can correctly classify all the remaining examples in the training set, when $\widehat{X}$ is used for a nearest neighbor (NN) rule. The idea behind this implementation of a consistent subset is to eliminate the examples from the majority class that are distant from the decision border, since these sorts of examples might be considered less relevant for learning. This method is only effective in data sets having less overlap; in case of high overlap between the class, no important reduction in the training set can be achievable.

Wilson's Edited Nearest Neighbor Rule (ENN) [23] ENN removes any example from the data sets whose class label differs from the class of at least two of its three nearest neighbors.

Neighborhood Cleaning Rule Neighborhood Cleaning Rule (NCL) [24] modifies the ENN in order to increase the data cleaning. For a two-class problem the algorithm can be described in the following way: for each example X_i in the training set, its three nearest neighbors are found. If X_i belongs to the majority class and the classification given by its three nearest neighbors contradicts the original class of X_i, then X_i is removed. If X_i belongs to the minority class and its three nearest neighbors misclassify X_i, then the nearest neighbors that belong to the majority class are removed. This method is only effective in data sets having less class overlap; in case of high overlap between the classes all majority class examples near to decision boundary will be eliminated and the training set will result in a poor model for the majority class.

Tomek links Tomek links[25] consider the examples near to the borderline to be more important. The method can be defined as follows: given two examples X_i and X_j belonging to different classes, and $d(X_i, X_j)$ the distance between X_i and X_j, a pair (X_i, X_j) is called a Tomek link if there is not an example X_l, such that $d(X_i, X_l) < d(X_i, X_j)$ or $d(X_j, X_l) < d(X_i, X_j)$. If two examples form a Tomek link, then either one of these examples is noise or both examples form the borderline. Tomek links can be used as an under-sampling method or as a data cleaning method. As an under-sampling method, all those examples which form Tomek link and belonging to the

majority class are eliminated, and as a data cleaning method, Tomek link examples from both classes are removed. This must be used with caution in highly imbalanced data sets in the presence of highly overlapped classes, as we may end up heavily reducing the majority class, hence the accuracy of majority class will be seriously affected.

One-sided selection (OSS) OSS [26] is an under-sampling method resulting from the application of Tomek links followed by the application of CNN. Tomek links is used as an under-sampling method and removes noisy and borderline majority class examples. Borderline examples can be considered as unsafe, since a small amount of noise can make them fall on the wrong side of the decision border. CNN aims to remove examples from the majority class that are distant from the decision border. The remaining examples, i.e. majority class examples and all minority class examples are used for learning.

CNN + Tomek links This is one of the methods proposed by Batista et al. [18]. It is similar to the OSS, but the method to find the consistent subset is applied before the Tomek links. Their objective is to verify its competitiveness with OSS. As finding Tomek links is computationally demanding, it would be computationally cheaper if it was performed on a reduced data set.

2.3.1.2 Random over-sampling

Random over-sampling involves duplicating examples of the minority class in order to achieve a more balanced distribution. Both under-sampling and over-sampling have known drawbacks: random over-sampling can increase the likelihood of over-fitting, since most over-sampling methods make exact copies of the minority class examples [7, 26]. There are several heuristic over-sampling methods, such as SMOTE, and various variations of the SMOTE algorithm.

Smote Synthetic Minority Over-sampling Technique (Smote) [7] is an over-sampling method. Its main idea is to form new minority class examples by interpolating between several minority class examples that lie close together. Thus, the overfitting problem is avoided and causes the decision boundaries for the minority class to spread further into the majority

class space, in cases of class overlap.

Realizing the importance of borderline examples, i.e., that these examples can be more easily misclassified than those far from the borderline, Han et al. [27] presented a modification of the SMOTE technique, which they named as **Borderline-SMOTE (BSM)**. BSM selects minority examples which are considered to be on the border of the minority decision region in the feature-space and only performs SMOTE to over-sample those examples, rather than over-sampling them all or a random subset. Their experiments show better accuracy of the classifier for the minority class than original SMOTE or random over-sampling.

Chawla et al. [28] suggested that boosting (see next section) may suffer from the same problem as over-sampling (i.e., over-fitting). Instead of changing the distribution of training data by updating the weights associated with each example they introduce SMOTEBOOST [28]. SMOTEBOOST alters the distribution by adding new examples of the minority class using the SMOTE algorithm.

2.3.1.3 Active Re-sampling Techniques

In comparison to random re-sampling techniques or neighborhood cleansing techniques, the following are examples from active sampling techniques or informed sampling.

Boosting is an iterative algorithm that places different weights on the examples of the training set at each iteration. The purpose of the Boosting algorithm [29] is to improve the classification accuracies of any "weak" learning algorithm. It weights each sample reflecting its importance and places more weight on those examples which are more often misclassified. This forces the learner to those samples that are hard to correctly classify. Since boosting effectively alters the training distribution it can be regarded as an active sampling technique.

Provost et. al [8] analyze methods for progressive sampling as long as model accuracy improves. Progressive sampling starts with a small sample and use progressively larger ones until model accuracy no longer improves.

They use a learning curve to decide when the model has converged (i.e., accuracy of the model does not change with increase of the training sets). A learning curve depicts the relationship between sample size and model accuracy. In the case of class imbalance this methodology can be adapted to using all the examples of minority class and a small subset of the majority class to build up the model, then at each iteration increasing the examples of the majority class until overall classification accuracy does not increase. Hence this technique can be used to build up an optimal class distribution, for a specific learning algorithm. But one of the drawbacks is that it is dependent on the classifier used, i.e., the optimal class distribution may not the optimum for another classifier.

Drown et al. [30] introduced Evolutionary Sampling Technique using the Genetic Algorithm (GA) [31] (GA is a random search method that can effectively explore large search spaces) to choose a reduced sample of the complete data set to train a classification model. As one-sided sampling (OSS) [26] and Wilson's Editing (WE) [23] attempt to improve under-sampling by nearest neighbor rule to determine which examples are redundant or noisy, Evolutionary Sampling tries to achieve the same goal using the Genetic Algorithm.

It should be noted that all the methods explained above are trying to fix between-class imbalances. To deal with the problems of both between and within class imbalances, Cluster-based over-sampling (CBOS) [32] attempts to even out the between-class imbalance as well as the within-class imbalance simultaneously. There may be subsets of the examples of one class that are isolated in the feature-space from other examples of the same class, creating a "within-class imbalance". Small subsets of isolated examples are called "small disjuncts". Small disjuncts often cause degraded classifier performance, and CBOS aims to eliminate them without removing data, i.e., it clusters the majority class into k groups, selects the cluster with highest number of observations, and balances the number of observations in remaining clusters of majority class by over-sampling in each cluster. Similarly minority class is clustered into k clusters and the number of the observations in each minority class cluster is over-sampled in proportion to total number of observations in majority class.

Yuan et al. [33] proposed an approach called Support Cluster Machines (SCMs). In their proposed approach, they first partition the majority class examples into disjoint clusters, then train an initial SVMs [34] model using the minority class examples and representatives (centroid of each cluster) of the majority class clusters. From the initial SVMs, they can approximately identify the support vectors and non-support vectors. A shrinking technique is then used to remove the examples which are not support vectors. This procedure of clustering and shrinking is performed iteratively several times until the number of majority class examples are not more than five times that of the minority class examples.

When the data is highly imbalanced, under-sampling and over-sampling can be combined to improve the performance of the classifier [7, 18, 27, 24]. Batista et al. [18] propose the combination of various re-sampling strategies, like Smote combining with Tomek links or Smote combining with ENN, for highly imbalanced data sets.

2.3.2 Cost-Sensitive Learning

Class imbalance and cost-sensitive learning are related to each other. The differences between different misclassification errors can be quite large. Cost sensitive learning solutions incorporating both the data and algorithmic level approaches assume higher misclassification costs for samples in the minority class and seek to minimize these high cost errors. A cost-sensitive learning system can be used in applications where the misclassification costs are known. Cost-sensitive learning systems attempt to reduce the cost of misclassified examples, instead of classification error. These methods allow for the fact that the value of correctly identifying the positive (rare) class outweighs the value of correctly identifying the common class. For two-class problems this is done by associating a greater cost with false negatives than with false positives. This strategy is appropriate for most medical diagnosis tasks because a false positive typically leads to more comprehensive (i.e., expensive) testing procedures that will ultimately discover the error, whereas a false negative may cause a life-threatening condition to go undiagnosed, which

could lead to death. Assigning a greater cost to false negatives than to false positives will improve performance with respect to the positive (rare) class.

If for example this misclassification cost ratio is 3:1, then a space that has ten majority class examples and four minority class examples will nonetheless be labeled as the minority class. Thus non-uniform costs can bias the classifier to perform well on the minority class, where in this case the bias is desirable. One problem with this approach is that specific cost information is rarely available. This is partially due to the fact that these costs often depend on multiple considerations that are not easily compared. For example, in the medical diagnosis task the considerations involve the probability that an undiagnosed condition will lead to death, the "cost" of a false positive on a patient's well being, etc. Thus, without specific cost information, it may be more practical to only predict the rare class and generate an ordered list of the best minority predicting rules. Then one can decide where to place the threshold after data mining is complete. In mathematical notation, let the (i, j) entry in a cost matrix C be the cost of predicting class i when the true class is j. If $i = j$ then the prediction is correct so cost of correct classification is zero, i.e., $C(i, i) = 0$, while if $i \neq j$ the prediction is incorrect. The optimal prediction for an example x is the class i that minimizes.

$$L(x, i) = \Sigma_j P(j|x)C(i, j) \tag{2.1}$$

For each i, $L(x, i)$ is a sum over the alternative possibilities for the true class of x. In this framework, the role of a learning algorithm is to produce a classifier that for any example x can estimate the probability $P(j|x)$ of each class j being the true class of x. For example making the prediction i means acting as if i is the true class of x. It is noteworthy that the outputs of the research on class-dependent cost-sensitive learning have been good solutions to learning from imbalanced data sets [7, 14]. Cost-Sensitive learning has been suggested as a good solution to these class-imbalance tasks, yet it is not clear how the class-imbalance affects the cost sensitive classifier.

Liu and Zhou [35] gave an empirical study for the influence of class imbalance on Cost-Sensitive Learning. From their experiment they conclude

that class imbalance often affects the performance of cost sensitive classifiers. Cost-sensitive classifiers generally favour the original class distribution when misclassification costs differ slightly, while a balanced class distribution is more favorable when costs differ seriously.

MetaCost [36] is another method to make the classifier cost-sensitive. This classifier is equivalent to passing the base learner to Bagging (see section 2.4), which is in turn passed to a cost-sensitive classifier operating on minimum expected cost i.e., Equation 2.1. The difference is that MetaCost produces a single cost-sensitive classifier of the base learner. It is a two phase procedure, where in the first stage, an internal cost sensitive model is learned using a base cost sensitive learning algorithm. In the second stage the MetaCost procedure estimates class probabilities using bagging, relabels the training examples with minimum expected cost classes, and finally rebuilds the model using the modified training set.

AdaBoost (see section 2.3.4) has been made cost sensitive [37], so that examples belonging to the minority class that are misclassified are assigned higher weights than those belonging to the majority class. The resulting system, Adacost, has been shown empirically to produce lower cumulative misclassification costs than Adaboost.

2.3.2.1 Tuning parameter for Classifiers

Drummond and Holte [38] reported that using classifier C4.5 [39] at its default setting, over-sampling is ineffective, often producing little or no change using modified cost sensitive technique or class distribution. Moreover they noted that over-sampling prunes less, therefore generalizes less, than under-sampling, and modification of C4.5's parameters to increase the influence of pruning and avoidance of other factor such as over-fitting, can improve the performance of over-sampling. Similarly Japkowicz and Stephen [40] argued that, for severely highly imbalanced data sets, unpruned C4.5 models are better than the pruned versions. Wu and Chang [41] proposed an algorithm for Support Vector Machine (SVM) by changing the kernel function to improve the accuracy of the minority class. Veropoulos et al. [42] proposed

that using a different penalty constant for different classes will improve the accuracy of the minority class. On the basis of these studies we can argue that tuning the classifier's parameter (detail is given in Appendix 2.7) can help in building better models for imbalanced data sets.

Kaizhu Hang et al. [43] presented Biased Minimax Probability Machine (BMPM) to solve imbalance problem. With reliable mean vectors and covariance matrices for the minority and majority class, BMPM can derive the decision hyper-plane by adjusting the lower bound of the real accuracy of the testing set.

2.3.2.2 One class learning

One class learning is recognition based learning, where a model can be created with examples from the target class only. This learning provides an alternative to discriminant analysis. One-class SVM [44] is an example of one-class learning, for which Manevitz and Yousef [44] found one-class SVM to be competitive with two-class learning. However they believe that the results from a classifier trained using only positive class examples will not be as good as the result using positive and negative class examples.

Kubat et al. [26] introduced a technique "SHRINK" to cope with the problem of imbalance. This technique is another example of one class learning. This system labels a mixed space (where both minority and majority class examples are found) as the positive (minority class) regardless whether the positive examples prevail in the region or not, which changes the learner's focus: Then it searches the best positive space, i.e., the one with the best positive to negative ratio.

Ripper [45] is a rule induction method that uses a divide-and-conquer approach to build rules on the training set iteratively. Each rule is grown by adding new conditions until no majority class examples are found. It normally generates rules for each class, so it can be viewed as one class learning.

Raskutti and Kowalczyk [46] compare one-class SVM and two class SVM, suggesting that one class learning is useful in the presence of extremely un-

balanced data sets composed of high-dimensional feature space. They argue that one-class learning is related to the feature selection method (see below), but is more practical as feature selection is too expensive to apply.

2.3.3 Feature Selection

The goal of feature selection is to select a subset of features that allow the classifier to reach optimal performance. Generally feature selection methods can be divided into two groups: wrapper-based and filter-based feature selections. For wrapper-based techniques a classifier is involved in feature selection. The major disadvantages of wrapper-based methods are high computational cost and risk of over fitting. Whereas, in filter-based selection techniques, inherent characteristics of the data are used to select the feature. Their major disadvantage is to ignore the interaction between the features, as this technique selects each feature independently of the other.

Using feature selection for imbalanced data sets is a recent development with the majority of the work focused on text classification and web categorization domains [47, 48]. Forman [47] investigated multiple filter-based feature ranking techniques.

Zheng at al. [49] observed that the existing measures used for feature selection are not appropriate for class imbalanced data sets. They proposed selecting features for positive and negative classes separately, and then explicitly combining them. They used a filter-based technique to create one-sided and two sided metrics; one sided metrics select only positive features on their scores, and two sided metrics select both positive and negative features based on the absolute value of their scores. They then compared the performance of one sided and two sided metrics with different ratios of best positive and negative features. The ratio selection method results in better performance, compared to one and two sided metrics. Thus both positive and negative features are important to build a model. Castillo and Serrano [50] also used feature selection as the part of their studies. They used Genetic Algorithm [31] along with a filter-based feature selection method.

2.3.4 Ensemble Classifiers

An ensemble classifier uses a collection of base classifiers, instead of one single classifier, to make predictions. Breiman [51] used Bagging one of the ensemble methods, to reduce the prediction error by 20% on average over various problems. Boosting and Bagging are the most successful approaches of ensemble classification. The general ensemble approaches still have the same problem (low classification accuracy of minority class) in class imbalance applications but have been extensively used to handle class imbalance problems.

The AdaBoost (Adaptive Boosting) algorithm [29, 52] is an effective boosting algorithm to improve classification accuracies of any "weak" algorithm. It increases the weight of each example which has been misclassified by the preceding classifier. This forces the following classifier to focus on those examples which are more often misclassified. The AdaBoost algorithm is stated to be immune to over-fitting [53]. Therefore, Boosting is an attractive technique in dealing with class imbalance problems. In the 2-class problems some variants of the AdaBoost have been reported: AdaCost [37], Cost-sensitive Boosting (CSB1 and CSB2) [54], AdaC1, AdaC2 and AdaC3 [55]. These algorithms use cost information, also regarded as cost-sensitive boosting algorithms. As Boosting still suffer from low classification accuracy of minority class, the following variants are proposed for boosting algorithm: SMOTEBOOST [28] instead of increasing weights of any misclassified class creates synthetic examples for minority class; RareBoost [56] uses comparison of True Positive (TP) and False Positive (FP) and True Negative (TN) and False Negative (FN) (TP, FP, TN and FN are explained in next section) in determining the weights of each misclassified example and RUSBoost [57] uses random under-sampling of majority class before boosting algorithm.

Chan and Stolfo [58] ran multiple experiments to find the ideal class distribution for building the training model and then used re-sampling to generate multiple training sets with the desired distribution. Each training set includes all the minority class examples and a subset of the majority class examples, where each majority class example is guaranteed to occur in

at least one training set, hence no information is lost. The classifier is applied on each data set and a meta-classifier is used to form a composite classifier from the resulting classification.

The same basic approach for partitioning the data into n subsets and learning multiple classifiers has been used in a study conducted by Yan et al. [59]. Yan et al. [59] investigate SVM (Support Vector Machine) ensembles built with partitioning and various techniques for combining n models to make the final classification decision. They propose hierarchical SVMs for aggregating the outputs of SVM ensembles.

Molinara et al. [60] investigate partitioning using two splitting methods: random splitting and clustering. Random splitting simply separates the majority class into n independent disjoint random partition, then n models are constructed each using one partition of the majority class and the entire minority class. For the clustering approach, they used k-means clustering to partition the majority class into k partitions. Each of the majority class partitions are combined with the entire minority class to create k training data sets. Random under-sampling or over-sampling is used in each of the k training sets to balance the class distribution. Their results showed that random splitting is superior to the cluster based approach. This is to be expected, since in k-clustering we can not control the number of majority class examples in each cluster, and we may find a number of clusters where the majority class examples are less than the minority class. Moreover k-means is highly dependent on the choice of k, which is still an open question. Other clustering approaches have been proposed to decompose complex decision boundary, e.g., non-linear separable boundary into linear separable boundary [61] and using clustering for under-sampling of the majority class [62, 63].

2.4 Evaluation Measures

Evaluation measures play a crucial role in both assessing the classification performance and guiding the classifier modeling. Traditionally, accuracy as define below is the most commonly used measure for these purposes. However, for classification with the class imbalance problem, accuracy is no longer

a proper measure since the rare class has very little impact on accuracy as compared to the prevalent class [14]. For example, in a problem where a rare class is represented by only 1% of the training data, a simple strategy can be to predict the prevalent class label for every example. It can achieve a high accuracy of 99%. However, this measurement is meaningless in some applications where the learning concern is the identification of the rare cases.

When dealing with a binary classification problem we can always label one class as a positive and the other one as a negative class. The test set consists of P positive and N negative examples. A classifier assigns a class to each of them, but some of the assignments are wrong. To assess the classification results we count the number of True Positive (TP), True Negative (TN), False Positive (FP) (actually negative, but classified as positive) and False Negative (FN) (actually positive, but classified as negative). TP is also called 'hit', whereas FP is called 'miss' [64]. In biomedical applications, TP/P is called 'sensitivity' whereas TN/N is called 'specificity' [65].

It holds that

$$TP + FN = P \tag{2.2}$$

and

$$TN + FP = N \tag{2.3}$$

The classifier has assigned $TP + FP$ examples to the positive class and $TN + FN$ examples to the negative class. Let us define a few well-known and widely used measures:

$$specificity = \frac{TN}{N} \Rightarrow 1 - specificity = \frac{FP}{N} = FPR \tag{2.4}$$

$$sensitivity = \frac{TP}{P} = TPR = recall \tag{2.5}$$

$$Yrate = \frac{TP + FP}{P + N} \tag{2.6}$$

$$precision = \frac{TP}{TP + FP} \tag{2.7}$$

$$accuracy = \frac{TP + TN}{P + N} \tag{2.8}$$

$$\text{missclassification error (MER)} = 1 - accuracy = \frac{FP + FN}{P + N} \tag{2.9}$$

Precision, recall and accuracy (or MER) are often used to measure the classification quality of binary classifiers. The FPR (false positive rate) measures the fraction of misclassified examples. The TPR (true positive rate) or recall measures the fraction of correctly classified examples. Precision measures that fraction of positive examples classified that are true. Lift tells how much better a classifier predicts compared to a random selection. It compares the precision to the overall ratio of positive values (Yrate) in the test set.

$$lift = \frac{Precision}{P/(P + N)} = \frac{Sensitivity}{Yrate} \tag{2.10}$$

2.4.1 Probabilistic Classifiers

Probabilistic classifiers assign a score or a probability to each example. A probabilistic classifier is a function $f : X \to [0, 1]$ that maps each example x to a real number $f(x)$. Normally, a threshold t is selected for which the examples with $f(x) \geq t$ are considered to be target example and the others are considered non-target examples.

This implies that each pair of a probabilistic classifier and threshold t defines a binary classifier. Measures defined in the section above can therefore also be used for probabilistic classifiers, but they are always a function of the threshold t. Note that TP(t) and FP(t) are always monotonic descending functions. For a finite example set they are stepwise, not continuous. By varying t we get a family of binary classifiers.

Note that some classifiers return a score between 0 and 1 instead of a probability. For the sake of simplicity we shall call them also probabilistic classifiers, since an uncalibrated score function can be converted to a proba-

bility function.

2.4.2 F-measure

As mentioned earlier, accuracy is not suitable to evaluate imbalanced data sets, as accuracy places more weights on the majority class than on minority class, which makes it difficult for a classifier to perform well on the minority class. Due to this reason, additional metrics are coming into widespread use.

F-value (or F-measure) is a popular evaluation metric for imbalance problems [66]. If only the performance of the positive class is considered, two measures are important: True Positive Rate (TPR) and Positive Predictive Value (PP value). In information retrieval, True Positive Rate is defined as recall denoting the percentage of retrieved objects that are relevant. Van Rijsbergen [67] defined the E-measure as a combination of Recall ($\underline{\mathrm{R}}$) and Precision ($\underline{\mathrm{P}}$) satisfying certain measurement theoretic properties:

$$E = 1 - \frac{(1+\beta^2)\underline{P}\underline{R}}{(\beta^2)\underline{P} + \underline{R}} \tag{2.11}$$

F-measure also depends on the β factor, a parameter that takes a value between 0 and infinity and correspond to relative importance of precision vs recall. It can be shown that when $\beta = 0$ then F-measure reduces to precision and conversely when $\beta \to \infty$ then F-measure approaches to recall. A β of 1 corresponds to equal weighting of recall and precision. To get a single measure of effectiveness where higher values correspond to better effectiveness, and where recall and precision are of equal importance, Lewis and Gale [68] define $F_{\beta=1} = 1 - E_{\beta=1}$.

F-measure (F) is suggested to integrate Recall and Precision as an average

$$\text{F-measure} = \frac{2\underline{R}\underline{P}}{\underline{R} + \underline{P}} \tag{2.12}$$

In principle, F-measure represents a harmonic mean between recall and precision

$$\text{F-measure} = \frac{2}{1/\underline{P} + 1/\underline{R}} \tag{2.13}$$

The harmonic mean of two numbers tends to be closer to the smaller of the two. Hence, a high F-measure value ensures that both recall and precision are reasonably high.

2.4.3 G-mean

When the performance of both classes is concerned, both True Positive Rate (TPR) and True Negative Rate (TNR) are expected to be high simultaneously. Kubat et al. [26] suggested the G-mean defined as

$$\text{G-mean} = \sqrt{TPR \times TNR} \tag{2.14}$$

2.4.4 Receiver Operating Characteristic (ROC)

Perhaps the most common metric to assess overall classification performance is Receiver Operating Characteristic (ROC) analysis and the associated use of the area under the ROC curve (AUC) [69]. When we want to assess the accuracy of a classifier independent of any threshold, ROC analysis can be used. In ROC space, one plots the False Positive Rate (FPR) on the x-axis and the True Positive Rate (TPR) on the y-axis. Some classifiers, such as Bayesian network inference or some neural networks, assign a probabilistic score to its prediction. Class prediction can be changed by varying the score threshold. Each threshold value generates a pair of measurements of (FPR, TPR). By linking these measurements with the False Positive Rate (FPR) on the X-axis and the True Positive Rate (TPR) on the Y -axis, an ROC graph is plotted, please see Fig 2.1. An ROC graph is defined by a parametric definition [70]

$$x = 1 - specificity(t), \quad y = sensitivity(t) \tag{2.15}$$

The ideal model is one that obtains 1 True Positive Rate and 0 False Positive Rate (i.e., TPR = 1 and FPR = 0). A model that makes a random guess should reside along the line connecting the points (TPR= 0, FPR = 0), where every instance is predicted as a negative class, and (TPR = 1, FPR

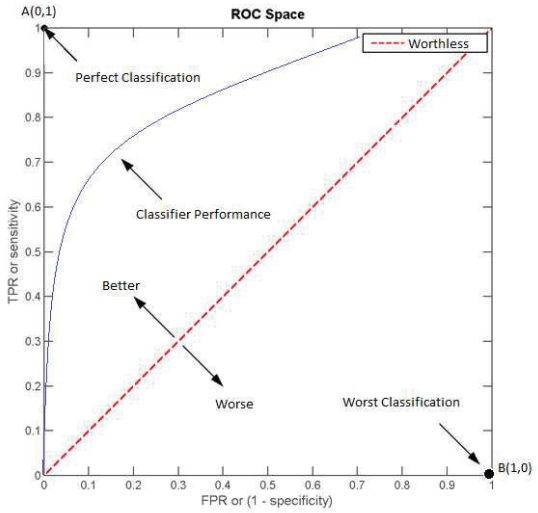

Figure 2.1: ROC Curve: the points on the curve represent the performance of the classifier. The ideal model is one that obtains a True Positive Rate of one and a zero False Positive Rate (i.e., TPrate = 1 and FPrate = 0, point A in figure. A worst case scenario would be point B, coordinates (1,0), where TPR is zero and FPR is a maximum. A model that makes a random guess should reside along the line connecting the points (TPrate = 0, FPrate = 0), where every instance is predicted as a negative class, and (TPrate = 1, FPrate = 1), where every instance is predicted as a positive class.

= 1), where every instance is predicted as a positive class. An ROC graph depicts relative trade-offs between benefits (true positives) and costs (false positives) across a range of thresholds of a classification model. An ROC curve gives a good summary of the performance of a classification model. To compare several classification models by comparing ROC curves, it is hard to claim a winner unless one curve clearly dominates the others over the entire space. The ROC representation allows an experimenter to see quickly if one classifier dominates another and therefore, using the convex hull, to identify potentially optimal classifiers visually without committing to a specific performance measure.

Each binary classifier (for a given test set of examples) is represented by a point (1-specificity, sensitivity) on the graph. By varying the threshold of

the probabilistic classifier, we get a set of binary classifiers, represented with a set of points on the graph. The ROC curve is independent of the P : N ratio and is therefore suitable for comparing classifiers when this ratio may vary. Note that the Precision-Recall curve can also be computed, but Davis and Goadrich [71] showed the curve is equivalent to the ROC curve.

Area under ROC curve is often used as a measure of quality of a probabilistic classifier. The area under an ROC curve (AUC) provides a single measure of a classifier's performance for evaluating which model is better on average. It has been shown in the literature that there is a clear similarity between AUC and well-known Wilcoxon statistics [72]. It is close to the perception of classification quality that most people have. AUC is computed with the following formula:

$$AUC = \int_0^1 \frac{TP}{P} d\frac{FP}{N} = \frac{1}{P.N} \int_0^N TP dFP \tag{2.16}$$

A random classifier (e.g. classifying by tossing a coin) has an area under curve of 0.5, while a perfect classifier has 1. Classifiers used in practice should therefore be somewhere in between, preferably close to 1. What does AUC really express? For each negative example count the number of positive examples with a higher assigned score than the negative example, sum it up and divide everything by $P \times N$. This is exactly the same procedure as used to compute the probability that a random positive example has a higher assigned score than random negative example.

$$AUC = P(Score_{\text{target}} > Score_{\text{non-target}}) \tag{2.17}$$

2.5 Discussion

It has been observed in studies like [17, 73], that in some domains, standard classifiers are capable of inducing a good model, even when using a highly imbalanced training set. Hence it can be argued that class imbalance may not be the only reason for poor accuracy of the minority class, but that there are

some other factors as well. Studies conducted by Japkowicz [74] suggested that not only class imbalance between classes but also within class imbalance is also relevant. Jo and Japkowicz [32] suggested that class imbalances may yield small disjunct, causing degradation. Similarly Prati et al. [11] argued that class imbalance in the presence of class overlap causes problem.

Hence there is general understanding that class imbalance in the presence of other factors such as small disjunct, within class imbalance or class overlap etc, is the cause of degradation in the accuracy of the minority class. However none of studies have been specifically design to tackle these problem, although there is some suggestion that using clustering techniques [32] attempts to even out the between-class imbalance as well as the within-class imbalance simultaneously. Clustering techniques in our opinion are too vague to apply, as different studies have resulted in mixed results using clustering techniques [60]. Moreover, none of studies found a particular classifier or learning algorithm that works consistently better in the class imbalance scenario. There are however some suggestions that re-sampling works better in increasing the accuracy of minority class with certain classifiers [17]. There are a great many re-sampling approaches proposed in the literature, that work better with some imbalanced data sets but not with every imbalanced data set, hence future research should examine this issue specifically.

2.6 Conclusion

The Class Imbalance problem is an important issue in the data mining and machine learning community. Our survey of techniques shows that there has been lot of research in this area especially from the last two decades. There seems to be a need for a structured study to delineate and understand the problems with imbalanced data. A deeper understanding of the data sets, through quantification of the relative degree of class learning difficulty among data sets, will help us to design better methods or to use existing methods in a better way for dealing with the class imbalance problem. There is lot of research in re-sampling or cost sensitive algorithms, but we find only a few studies that use feature selection in class imbalance problems. Hence there is

need for more research in this field, specifically in class imbalance situations: to design feature selection methodology that suits class imbalance scenarios.

References

[1] Kubat, M., Holte, R. C., Matwin, S., Machine learning for the detection of oil spills in satellite radar images, Machine Learning, Vol. 30, no. 2-3, pp. 195-215, 1998.

[2] Fawcett, T. and Provost, F., Adaptive fraud detection, Data Minining and Knowledge Discovery, Vol. 1, no. 3, pp. 291-316, 1997.

[3] Hilas, C. S. and P.A. Mastorocostas, P. A., An application of supervised and unsupervised learning approaches to telecommunications fraud detection, Knowledge-Based Systems, Vol. 21, no. 7, pp. 721-726, 2008.

[4] Estabrooks, A., Jo, T. and Japkowicz, N., A Multitple Resampling Method For Learning From Imabalanced Data Sets. Computational Intelligence, Vol. 20, no. 1, pp. 18 - 36, 2004.

[5] Holte, R. C., Acker, L. E. and Porter, B. W, Concept learning and the problem of small disjuncts. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pp. 813-818, 1989.

[6] Japkowicz, N., Class imbalances, Are we focusing on the right issue?, Proceeding of ICML-2003, Workshop: Learning with Imbalanced Data Sets II, pp. 17-23, 2003.

[7] Chawla, N., Bowyer, K.,Hall, K., Kegelmeyer, P., SMOTE: Synthetic Minority Over-Sampling Technique, Journal of Artificial Intelligence Research, Vol. 16, pp. 321-357, 2002.

[8] Provost, F., Jensen, D., and Oates, T. Efficint Progressive Sampling. Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, San Diego, California, United States, pp. 23-32, 1999.

[9] Elkan, C., The foundation of cost sensitive learing, Proceeding 17th International joint conference for Artificial Intelligence, pp. 973-978, 2001.

[10] Batista,G.E., Prati, R.C., and Monard,M.C., Balancing strategies and class overlapping. In proceeding of 6th International Symposium on Intelligent Data Analysis, Madrid, Spain, pp.24-35, 2005.

[11] Prati, R. C., Batisita, G. E., and Monard, M. C., Class Imbalances versus Class Overalpping, An Analysis of the Learning System Behaviour, in MICAI 2004, pp. 312-321, 2004.

[12] Vias, S. and Ralescu, A., Issue in Mining Imbalanced Data sets- A Review Paper, in Proceedings of the Sixteen Midwest Aritficial Intelligence and Cognitive Science Conference, MAICS, pp. 67-73, 2005.

[13] Kotsiantis, S., Kanellopoulos, D., and Pintelas, P., Handling imbalanced datasets: A review, GESTS Interantional Transaction on Computer Science and Engineering , Vol. 30, no. 1, pp. 25-36, 2006.

[14] Weiss, G.M., Mining with Rarity: A Unifying Framework. SIGKDD Explorations, Vol. 6, no. 1, pp.7-19, 2004.

[15] Proceeding of AAAI'00, Workshop Learning from Imbalanced Data sets, Japkowicz, N., ed., 2000.

[16] ICML '03, Workshop Learning from Imbalanced Data sets, Chawal, N., Japkowicz, N. and Kolcz, A., 2003.

[17] Hulse, J.V., Khoshgoftaar, T.M., Napolitano, A., Experimental perspectives on learning from imbalanced data, in: Proceedings of the 24th International Conference on Machine Learning, Corvalis, Oregon, pp. 935-942, 2007.

[18] Batista,G.E., Prati, R.C., and Monard,M.C., A study of the behavior of several methods for balancing machine learning training data. IACM SIGKDD Explorations Newsletter, Vol. 6, no. 1, pp. 20-29, 2004.

[19] Garcia, V., Mollineda, R., Sanchez, J., S., On the k-NN performance in challenging scenario of imbalance and overlapping. Pattern Analysis Application Vol. 11, no. 3-4, pp. 269-280, 2008.

[20] Drummond, C. and Holte, R., Severe Class Imbalance: Why Better Algorithms Arent the Answer. In Proceedings of the 16th European Conference on Machine Learning (ECML/PKDD'05), pp. 539-546, 2005.

[21] Japkowicz, N., Learning from Imbalanced Data Sets, Proceeding of AAAI'00, 2000.

[22] Hart, P. E., The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory IT-14, pp. 515516, 1968.

[23] Wilson, D. L., Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Transactions on Systems, Management, and Communications, pp. 2(3), pp. 408-421, 1972.

[24] Laurikkala, J., Improving Identification of Difficult Small Classes by Balancing Class Distribution. Technical Report A-2001-2, University of Tampere, 2001.

[25] Tomek, I., Two Modifications of CNN. IEEE Transactions on Systems Man and Communications SMC-6, pp. 769-772, 1976.

[26] Kubat, M., and Matwin, S. Addressing the Curse of Imbalanced Data Sets: One Sided Sampling. In the Proceedings of the Fourteenth International Conference on Machine Learning, Nashville, TN, pp. 179- 186, 1997.

[27] Han, H., Wang, W., Mao, B., Border-SMOTE: A New Over-Sampling in Imbalanced Data Sets Learning. LNCS Springer, Heidelberg, pp. 878-887, 2005.

[28] Chawla, N. V., Lazarevic, L., Hall, L. O., and Bowyer, K. W., SMOTEBOOST: Improving prediction of the minority class in boosting, in preceeding of the seventh European Conference on Principles and Practice of Knowledge Discovery in Databases, Croatia, pp.107-119, 2003.

[29] Freund, Y. and Schapire, R., Experiments with a new boosting algorithm, in proceeding of the Thirteenth Intenational Conference on Machine Learning, pp. 148-156, 1996.

[30] Drown, D. J., Khoshgoftaar, T. M. and Narayanan, R., Using Evolutionary Sampling to Mine Imbalanced Data. In Proceedings of 4th international conference on Machine learning and Application, pp. 363-368, 2007.

[31] Holland, J. , Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, Michigan, 1975.

[32] Jo, T., Japkowicz, N. Class imbalances versus small disjuncts. SIGKDD Explorations, Vol. 6, pp.429-450, 2004.

[33] Yuan, J., Li, J.,and Zhang, B., Learning Concepts from Large Scale Imbalanced Data Sets Using Support Cluster Machines. In Proceedings of the 14th annual ACM international conference on Multimedia, Santa Barbara, California,USA, pp. 441-450, 2006.

[34] Vapnik, V., The Nature of Statistical Learning Theory, Springer Verlag, 1995.

[35] Liu, XY. and Zhou, ZH., The influence of class imbalance on cost-sensitive learning: An Empirical Study. In Proceeding 6th International Conference on Data Mining, pp. 970-974, 2006.

[36] Domingos, P., MetaCost: A general method for making classifier cost sensitive, in Proceeding of the Fifth International Conference on Knowledge Discovery and Data mining, ACM Press, pp. 155-164, 1999.

[37] Fan, W. Stolfo, S. J. and Chan, P. K., AdaCos: misclassification cost-sensitive boosting", in Proceeding of the Sixteenth International Conference on Machine Learning, pp. 97-122, 1999.

[38] Drummond, C. and Holte, R., C4.5, Class Imbalance, and Cost Sensitivity: Why Under-sampling beats Over-sampling. Workshop on Learning from Imbalanced Datasets II, ICML, Washington DC. 2003.

[39] Quinlan, J. R., C4.5: Programs For Machine Learning. Morgan Kaufmann, San Mateo, California, 1993.

[40] Japkowicz, N., and Stephen, S.: The Class Imbalance Problem: A Systematic Study. Intelligent Data Analysis Journal, Vol. 6: pp. 429-449, 2002.

[41] Wu, G. and Chang, E., Class-Boundary Alignment for Imbalanced Dataset Learning, In ICML 2003 Workshop on Learning from Imbalanced Data Sets II, Washington, DC, 2003.

[42] Veropoulos, K., Campbell, C., and Cristianini, N., Controlling the sensitivity of support vector machines, in Proceeding of the International Joint Conference on Artificial Intelligence, pp. 55-60, 1999.

[43] Huang, KZ., Yang, HQ., King, I., and Lyu, MR., Learning Classifier from Imbalanced Data Based on Biased Minimax Probability Machine, in Proceeding of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004.

[44] Manevitz, L. M. and Yousef, M., One SVMs for document classification, Journal of Machine Learning Research, Vol. 2, pp.139-154, 2001.

[45] Cohen, W. W., Fast effective rule induction. In Proceeding of the Twelfth International Conference on Machine Learning (ICML), pp.115-123, 1995.

[46] Raskuttis, B. and Kowalcyzyk, A., Extreme rebalancing for SVMs: a case study, SIGKDD Explorations, Vol. 6, no. 1, pp. 60-69, 2004.

[47] Forman, G.and Cohen, I., Learning from Little: Comparison of Classifiers Given Little Training, Proceeding 8th European conference, principles and practice of knowledge Discovery in Database, pp.161-172, 2004.

[48] Mladenic, D., and Grobelnik, M., Feature Selection for Unbalanced Class distribution and Naive Bayes, Proceeding of 16th international conference on Machine learning, pp. 258-267, 1999.

[49] Zheng, Z., Wu, X., and Srihari, R., Feature Selection for Text Categorization on Imbalanced Data, ACM SIGKDD Explorations Newletter, Vol. 6, pp. 80-89, 1999.

[50] Castillo, M., and Serrano, J, A multistrategy approach for digital text categorization from imbalanced document, SIGKDD Explorations, Vol. 6, no. 1 pp. 70-79, 2004.

[51] Breiman, L., Bias, Variance and arcing classifiers, Technical Report 460, Statistics Department, University of California, Berkeley, CA., 1996.

[52] R. E. Schapire, R. E. and Singer, Y., A decision-theoretic generalization of on-line learning and an aplication to boosting. Journal of Computer and System Sciences, Vol. 55, no. 1, pp.119-139, 1997.

[53] Friedman, J., Hastie, T., and Tibshirani, R., Additive logistic regression: a statistical view of boosting, Annals of Statistics, Vol. 28, no. 2 pp. 337-374, 2000.

[54] Ting, K.M., A Comparative study of cost-sensitive boosting algorithms, In proceeding of the 17th International Conference of Machine Learning, Stanford University, pp. 983-990, 2000.

[55] Sun, Y., Wong, A. K. C., and Wang, Y., Parameter inference of cost sensitive boosting algorithms. In proceedings of 4th International Conference on Machine Learning and Data Mining in Pattern Recognition, Germany, pp. 21-30, 2005.

[56] Joshi, M. V., Kumar, V., and Agarwal, R. C., Evaluation boosting algorithms to classify rare class: Comparison and improvements. In proceeding of the first IEEE international conference on Data Mining, 2001.

[57] Seiffert, C., Khooshgortaar, T. M., Hulse, Van, J. adn Napolitano, A., RUSBoost: Improving classification when training data is skewd, the 19th IEEE International Conference on Pattern Recognition, Tampa, Florida, USA, pp, 1-4, 2008.

[58] Chan, P.K., and Stolfo, S. J., Towards scalable learning with non-uniform class and cost distribution: a case study in credit card fraud detection, in Proceeding of the Fourth International conference on Knowledge Discovery and Data Mining, pp. 164-168, 2001.

[59] Yan, R., Liu, Y., Jin, R., and Hauptmann, On predicting rare classes with SVM ensembles in scence classification, in IEEE International Conference on Acoustics, Speech and Signal Processing 2003.

[60] Molinara, M., Ricamato, T. and Tortorella, F., Facing imbalanced classes through aggregation of classifiers, in Proceeding of the 14th International Conference of Image Analysis and Processing, Modena, Italy, IEEE Computer Society, pp. 43-48, 2007.

[61] Wu, J., Xiong, H., Chen, J., COG: local decomposition for rare class analysis, Data Mining and Knowledge Discovery, Springer Netherlands Vol. 20, no. 2, , pp. 191-220, 2010.

[62] Altincay, H. and Ergün, C., Clustering based under-sampling for improving speaker verification decisions using AdaBoost, Joint IAPR International Workshops on Structural, Syntactic, and Statistical Pattern Recognition. Lecture Notes in Computer Science, pp. 698-706, 2004.

[63] Yen, S-J. and Lee, Y-S, Under-Sampling Approaches for Improving Prediction of the Minority Class in an Imbalanced Dataset, In Proceedings of the Intelligent Control and Automation, Lecture Notes in Control and Information Sciences (LNCIS), Vol. 344, pp. 731-740, 2006.

[64] Chan, P.K. and Stolfo, S.J., Towards Scalable learning with non-uniform class and cost distribution. A case study in credit card fraud detection, in proceeding of the foruth Conference on Knowledge Discovery and Data Mining, New York, pp. 164-168, 1998.

[65] Chen, J.J., Tsai, C.A., Young, J.F. and Kodell, R. L., Classification ensembles for unbalanced class size in predictive toxicology. SAR and QSAR in Envoirmental Researchx, Vol. 16, no. 6, pp. 517-529, 2005.

[66] Estabrooks and Japkowicz, N., A mixture-of-experts framework for learning from unbalanced data sets, In proceeding of the 2001 Intelligent Data Analysis Conference, pp. 1145-1159, 2001.

[67] Rijsbergen, C. J. V., Information retrieval. Butterworths, London, 1979.

[68] Lewis, D. D. and Gale, W. A., A sequential algorithm for training text classifiers. In the Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval; pp. 312, 1994.

[69] Bradley, A.P., The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognition, Vol. 30, no. 7, 1997, pp.1145-1159.

[70] Fawcett, T., ROC Graphs: Notes and Practical Considerations for Data Mining Researchers. HP Laboratories, (2003).

[71] Davis, J. and Goadrich, M., The Relationship Between Precision-Recall and ROC Curves. In the Proceedings of the 23rd International Conference on Machine Learning (ICML), 2006.

[72] Hanley, J. A. and McNeil, B. J., The meaning and use of the area under a receiver operating characteristic (ROC) curve, Radiology, Vol. 143, pp.29-36, 1982.

[73] G. Weiss, G., McCarthy, K. and Zabar, B., Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs? In Proceedings of the 2007 International Conference on Data Mining; pp.35-41, 2007.

[74] Japkowicz, N., Comcpet-learning in the presence of between-class and within-class imblanaces" in proceedings of the fourteenth conference of Canadian Society for Computational Studies for intelligence, pp. 67-77, 2001.

[75] Venables, W.N. and Ripley, B.D., Modern Applied Statistics with S-PLUS, Springer, NewYork 1999.

[76] Therneau, T. M., Atkinson B., and Report by Brian Ripley rpart: Recursive Partitioning. R package version 3.1-42 (http://www.mayo.edu/biostatistics), 2004.

[77] Breiman, L. et al. Classifications and Regression Trees, Wadsworth and Brooks, Belmont, 1984.

[78] C.M. Bishop, C. M., Neural Networks for Pattern Recognition, Oxford University Press, NewYork, 1995.

[79] B.E. Boser, B.E., Guyon, I.M., and Vapnik, V.N., A training algorithm for optimal margin classifiers. In proceeding of the Fifth Annual ACM Workshop on Computational Learning Theory, Pittsburgh: ACM Press, pp. 144-152, 1992.

[80] Dimitriadou, E., et al., e1071: Misc Functions of the Department of Statistics (e1071), TU Wien. R package version 1.5-1 (http://www.r-project.org), 2004.

[81] Furey T.S., N. Christianini, N. Duy, D.W. Bednarski, M. Schummer, D. Haussler: Support vector machine classication and validation of cancer tissue samples using microarray expression data. Bioinformatics, Vol. 16, no. 10, pp.906-914, 2000.

2.7 Appendix "Classification Methods"

2.7.1 Logistic regression

For a binary response and p quantitative predictors $x_1, ..., x_p$, (some of them may be dummy variables for coding qualitative variables), the LR model assumes that the probability of the target response is

$$\pi(x_1, ..., x_p) = \frac{e^{\beta_0+\beta_1 x_1+...+\beta_p x_p}}{1 + e^{\beta_0+\beta_1 x_1+...+\beta_p x_p}} \tag{2.18}$$

The glm function in R (Venables and Ripley [75]) tries to compute the maximum likelihood of the $p + 1$ parameters by an iterative weighted least squares (IWLS) algorithm. There are several inferential procedures to test the statistical significance of the whole model and the individual significance of each variable. The model may also be interpreted and a great family of diagnostics criteria are available to identify influential and outlying observations. LR can be fully embedded in a formal decision framework but, in order to realize a comparison with the other models, including measures such as the accuracy, we have the necessity of specifying a threshold probability to assign a case to one of the two classes, which corresponds to varying the prior classes probabilities. Given a fixed threshold probability pc, the classification Y of a vector x will be classified as positive (1) or negative (0) through the rule:

$$\begin{bmatrix} 1 & \widehat{\pi}(x) \geq p_c \\ 0 & \widehat{\pi}(x) < p_c \end{bmatrix}$$

We will considered possible values for pc: 0.05, 0.10, 0.15, . . . ,0.95. The logistic regression model will be fitted to each training set.

2.7.2 Classification Trees

A classification tree (CT) is a set of logical if-then conditions, which drive each instance to a final decision. These conditions can be easily plotted helping us to understand the model. A binary CT is grown by binary recursive partitioning using the response in the specified formula and choosing splits

from the set of predictor variables. The split that maximizes the reduction in impurity (a measure of diversity for the outcome in a specific set of nodes) is chosen, the data set is split and the process is repeated. Splitting continues until the terminal nodes are too small to be split. The classification for a vector is computed by a majority class vote in its terminal node. We used the rpart package of R (Therneau and Atkinson [76]), which implements the CART methodology as proposed by Breiman et al. [77]. We used the Gini index (default impurity measure) as the splitting criterion. Given that large trees can lead to over fitting the data, with a loss in the generalization capability for new data, the user must tune a fundamental parameter (pruning): the number of terminal nodes, called the size of the tree. The rpart function computes, for each size, both the cross-validated (10-fold) estimate of the error and a standard deviation for error. The 1sd rule advises one to choose the smallest tree whose cross validated estimated error is less than minimum error+1 standard deviation. All of these measures are available from the R objects computed by the rpart function.

2.7.3 Multilayer perceptron

Artificial Neural Networks (ANN) are a computational paradigm providing a great variety of mathematical nonlinear models, and are useful for tackling different statistical problems. Several theoretical results support a particular architecture, namely the multilayer perceptron (MLP); for example, the universal approximate property, as in Bishop [78]. Following this result, we considered a three-layered perceptron with the logistic activation function $g(u) = \widehat{eu}/\widehat{eu} + 1$ in the hidden layer and the identity function as the activation function for the output layer. Denoting by H the size of the hidden layer, $v_{ih}, i = 0, 1, 2, ..., p, h = 1, 2, ..., H$ the synaptic weights for the connections between the p-sized input and the hidden layer, $w_{hj}, h = 0, 1, 2, ..., H, j = 1, 2, ..., q$ the synaptic weights for the connections between the hidden and the q-sized output layer, the outputs of the neural network are

$$o_j = w_{0j} + \sum_{h=1}^{H} \omega_{hj} g\big(v_{0h} + \sum_{i=1}^{p} v_{ih} x_i\big), \quad j = 1, 2, ..., q$$

In our classification problem the response was codified by a vector $z = (z_1, z_2)$ formed by two dummy variables 0 and 1, one for each class, so the number of outputs was $q = 2$. For an input vector to the net, the classification was that class corresponding to the dummy variable achieving the maximum of the two predictions of the net. An estimation of the probability of the class j, $j = 1, 2, ..., q$ is easily computed with the softmax expression:

$$\hat{P}[j] = \frac{e^{oj}}{\sum_{r=1}^{q} e^{or}}$$

The nnet R function (Venables and Ripley [75]) fits single-hidden-layer neural networks by the BFGS procedure, a quasi-Newton method also known as a variable metric algorithm, published in 1970 by Broyden, Fletcher, Goldfarb and Shanno, which tries to minimize a least squares criterion that allows a decay term intending to avoid over fitting problems. The BFGS algorithm can be found in Bishop [78]. Defining W as the vector of all the M coefficients of the net, the BFGS method is applied to the following non-linear least squares problem:

$$\min_W \sum_{i=i}^{n} \|Z_i - \widehat{z}_i\| + \lambda(\sum_{i=i}^{M} W_i^2)$$

A major disadvantage of MLP is the fact that there are no known procedures assuring us to obtain a global solution, and usually at most one of the many possible local minima is obtained. The R implementation of a MLP model requires the specification of two parameters: the size of the hidden layer (H) and the decay parameter (λ), and therefore we fix H=4 and decay(λ) = 0.01. In future we will search over a grid. defined as $5, 10, 15 \times 0, 0.01, 0.05, 0.1, 0.2, ..., 1.5$.

2.7.4 Support vector machines

Support Vector Machines (SVM) are a family of supervised machine learning techniques. They were originally introduced by Vapnik and other co-authors (Boser et al. [79]) and several extensions were successively proposed. When used for a two-class classification problem where the set of binary labeled training patterns is linearly separable, the SVM separates both classes with a

hyperplane that is maximally distant from them ('the maximal margin hyperplane'). If linear separation is not possible, the feature space is enlarged using basis expansions such as polynomials or splines. However, explicit specification of this transformation is not necessary, but a kernel function that computes inner products in the transformed space is required. We have fitted the SVM models with the svm function available in the library e1071 of the R system (Dimitriadou et al. [80]), which offers an interface to the award-winning C_{++} implementation, LIBSVM, by Chan & Lin. The data set is described by n training vectors $x_i, y_i, i = 1, 2, ..., n$ where the p-dimensional vectors x_i contain the predictor features and the n labels $y_i \epsilon \{-1, 1\}$ identify the class of each vector. Among the several variants of SVM existing in the R library e1071, following Meyer (2004) we have used C-classification with the Radial Basis Kernel because of the small number of parameters to be tuned (only two). The primal quadratic programming problem to be solved is:

$$\min_{\omega,b,\xi} \frac{1}{2}\omega^t\omega + C\sum_{i=1}^{n}\xi \; i = 1, 2, ..., n$$

$C > 0$ is a parameter controlling the trade-off between margin and error, and $\sum_{i=1}^{n}\xi$ is an upper bound on the sum of distances of the wrongly classified cases to their correct plane. The dual problem is

$$\min_{\alpha} \frac{1}{2}\alpha^t Q\alpha - e^t\alpha \quad 0 \leq \alpha \leq C$$
$$y^t\alpha = 0$$

where e is the n-vector of all ones, and Q is a positive semi-definite matrix defined by $Q_{ij} = y_i y_j K(x_i, x_j), i, j = 1, 2, ..., n$, being $K(x_i, x_j) = \phi(x_i)\phi(x_j)$ the kernel function. No explicit construction of the nonlinear mapping $\phi(x)$ is needed, what is called the kernel trick. A vector x is classified by the decision function

$$sign(\sum_{i=1}^{n} y_i\alpha_i K(x_i, x) + b)$$

depending on the margins $m_i = \sum_{i=1}^{n} y_i\alpha_i K(x_i, x) + b, \; i = 1, 2, ..., n$. The greater the absolute value of the margin, the more reliable is the computed classification. The exploratory analysis of the margins could help to enlighten the SVM model (Furey et al. [81]). The Radial Basis Function (RBF) was our choice for K:

$$K(u, v) = \exp(-\gamma \|u - v\|^2)$$

Note that the solution to the quadratic programming problem is global, avoiding the non optimality of the neural network training algorithms. So, two parameters must be tuned: C and γ. An estimation of the probability of both classes can be computed with the following expressions:

$$\widehat{P}[Class\ labeled\ \text{“0”}] = \frac{1}{1+e^{m_i}};\ \widehat{P}[Class\ labeled\ \text{“1”}] = \frac{e^{m_i}}{1+e^{m_i}}$$

Currently we used default value for the parameter C and γ of the svm function in the R library e1071, defined as $1/p$, with p being the number of predictors, while $C = 1$.

Chapter 3

Measurement and Visualization of Data Complexity for Classification Problems with Imbalanced Data

We introduce a complexity measure for classification problems that takes account of deterioration in classifier performance due to class imbalance. The measure is based on k-nearest neighbors. We explore the choices of k and the distance metric through a simulation study, and illustrate the use of our measure, and related data visualization techniques, with real data sets from the literature.

3.1 Introduction

Theoretically [1] and empirically, there is general understanding that different types of data require different kinds of classification. Not every imbalanced data set can be a problem for learning [2, 3]. It seems that the problem is rather with the small class in the presence of other factors such as class overlap [4]. The experiment conducted by Jo and Japkowicz [2] also suggested that degradation in class learning is not directly related to class imbalance

alone, but rather to small disjunct (small subsets of isolated examples).

Given the suggestion that the imbalanced class distribution may not be the only problem, we see the need for a structured way of investigating and explaining what intrinsic features of the data are affecting the degraded learning performance of an imbalanced data set. We think that the answer could be found through data complexity analysis or exploratory data analysis (EDA), and that data complexity measures can be used to devise a structured study for learning about class imbalance problems.

The classification error of an optimum Bayes classifier is an important parameter for pattern classification (the science of making inferences from perceptual data) and feature selection [5]. This parameter is recognized as a benchmark for other classification techniques and also as a "goodness" criterion for feature selection used in the classification. The Bayes error provides the lower error bound that can be achieved by any pattern classifier [6, 7]. This error rate will be greater than zero whenever class distributions overlap. When all the class priors and conditional likelihoods are completely known, in theory the Bayes rate can be obtainable [7], but only in simple situations and for Gaussian distributions can it be calculated directly. To measure the divergence between two Gaussian distributions the statistical literature suggests: Mahalanobis distance, Euclidean distance (most popular), J-Coefficient, Kullback-Leibler divergence, χ^2 divergence, Minkowsi L2 distance, Hellinger coefficient and Bhattacharyya distance. For more details see H.-H. Bock [8], Chapter: Dissimilarity Measures for Probability Distributions. However when the pattern distributions are unknown, the Bayes error cannot be readily computed. Thus, one cannot know how much classification error is due to class density overlap and how much is due to limitations of the training data (such as class imbalance) or deficiencies in the classifier. It is important to not only design a good classifier but to have a limit or bound on the achievable classification rate for a given data set [9]. Such an estimate will help researchers to decide whether to improve the current classifier, to use another classifier on the same data set or to acquire more data.

In the literature data complexity measures have been reviewed by Ho [10], who composed different complexity methods and was able to uncover

different data structures in real data sets, observing a relationship between the overall classification performance and data complexity. Details of these measures are given below. Another work by Weng and Poon [11] also looked at data complexity and tried to analyze classifier performance using the data complexity measurements of Ho [10]. They looked at just two data sets so did not do any correlation analysis to make the relation more explicit.

In this chapter we utilize data complexity to investigate the relationship between the degradation of classifier performance and certain features of the training data, specifically class imbalance and class overlap. Unlike alternative performance measures such as G-Mean and Sensitivity, our complexity is independent of the choice of the classifier. Data complexity in our context is the quantification of the degree of difficulty in class learning from a given data set. In this study we investigate class imbalance with theoretical known Bayes error. Our approach is also motivated geometrically in that we are looking at the location of cases as data points in a suitable space and how they are distributed within each class. This approach leads naturally to the consideration of data visualization as an additional tool for examining data complexity. We propose a new complexity measure designed to be sensitive to class imbalance, and investigate its properties and uses.

In the next section we review existing complexity measures. We then introduce our proposed measure based on k-Nearest Neighbors for estimation of Bayes error, and discuss visualization methods. Section 3.4 compares the properties of several complexity measures via a simulation study. In section 3.5 we illustrate our methodology using real data sets from the literature. We conclude with a discussion and suggestions for further research.

3.2 Data Complexity

In the literature a number of different but related approaches have been taken to measure the complexity of data sets. In this section we will review the current study on data complexity.

3.2.1 Studies on Data Complexity

Data complexity can be described as an indicator that shows the level of difficulty in class learning for a specific data set. Ho and Basu [12] proposed some complexity measures for binary classification problems. They classified complexity measurement under three different headings: overlaps of feature spaces from different classes, separability of classes and measures of geometry, topology and density of manifolds. This study has been widely used subsequently. In the literature we can see a number of studies using data complexity as a measure to define the strength of their methodology or to make a comparison between different classifiers using one or more of the complexity measures introduce by Ho and Basu [12]. For example studies conducted by Bernado-Mansilla and Ho [13], Li et al. [14], Baumgartner and Somorjai [15] and Garcia et al. [16] used complexity measures. They were not dealing with imbalanced data sets. Sanchez et al. [17] however compared different classifiers using artificial data sets with different degrees of imbalance, but no real data. Hence their study cannot infer an explicit relationship between data complexity and classifier performance in real data sets in imbalance cases. The core of the question is how some of the data complexity measures are related to the accuracy of the classification algorithms, and which data complexity measures appear to better describe the behavior of a classifier. Our aim is to investigate the utility of these measures in class imbalance scenarios.

3.2.2 Data Complexity Measures

Data complexity measures used in Ho and Basu [12] are summarized in Table 3.1.

In our simulation study, where the overlap is controlled and all dimensions are, by construction, equally important, we will use N1 and N2 to measure the separability of the classes. In our analysis for real data sets only F1, L3 and N1 measure of the 12 presented in Table 3.1 proved to be informative following the methodology explain in section 3.5. These measures are explained below.

Table 3.1: Data Complexity Measures by Ho and Basu

Groups	ID	Details
Measure of overlap in feature values from different classes	F1	Maximum Fisher discriminant ratio
	F2	Volume of overlap region
	F3	Maximum (Individual) feature efficiency
Measure of separability of Classes	L1	Minimized sum of error distance by linear programming
	L2	Error rate of linear classifier by linear programming
	N1	Fractions of points on class boundary
	N2	Ratio of average intra/inter class NN distance
	N3	Error rate of 1NN classifier
Mearsures of geometry, topology	L3	Nonlinearity of linear classifier by linear programming
and density of manifolds	N4	Non-linearity of 1NN classifier
	T1	Fraction of points with associated adherence subsets retained
	T2	Average number of points per dimensions

N1: Fractions of points on class boundary: The use of this measure was inspired by the test proposed by Friedman and Rafsky [18] for whether two multivariate samples are from the same distribution. This is given as the percentage of points on an edge connecting two opposite classes in the Minimum Spanning Tree (MST) connecting all training samples. In simple terms it is a measurement obtained by counting the number of boundary points, i.e. each of these points is a case connected to an example from a different class. The count is normalized as a proportion of total cases to give a value between 0 and 1: a value close to 0 means the data is separable and vice versa. According to Ho [10] this measure is sensitive to both the separability of the classes and the clustering tendency of the points of each class.

N2: Ratio of intra- versus inter-class nearest neighbors: This measure compares the dispersion within each class to the dispersion between the classes. In other words it compares the average distances between cases within each class to those in different classes. To compute this measure, first measure the average distance between points within a class

(intra-class) and the average distance from each point to its neighbors outside the class (inter-class). Dividing the intra-class average distance by the inter-class average distance, we get a ratio. A value close to zero indicates that cases are well differentiated into classes, and if the value is near to, or greater than, 1 it indicates that cases are highly overlapped.

F1: Maximum Fisher's discriminant ratio: A classical measure of the discriminative power of the covariates, or features, is Fisher's discriminant ratio:

$$f = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{3.1}$$

where μ_1, μ_2, σ_1^2 and σ_2^2 are the means and variances of the two classes, respectively. For multidimensional problems, it is not necessarily the case that all features contribute to class discrimination, so the maximum f over all features can be used. However, a zero maximum f does not necessarily mean that the classes are not separable, as it could just be that the separating boundary is not parallel to an axis in any of the given features [10].

L3: Nonlinearity of nearest neighbor or linear classifier: Hoekstra and Duin [19] proposed a nonlinearity measure for a classifier with respect to a data set. The measure is computed by randomly choosing pairs of points from the same class and creating an artificial case between them by linear interpolation. Then apply a nonlinear classifier on the artificial point and on one of the original pair of cases. If the classifier (e.g. a support vector machine with the linear kernel) disagrees on the two points then this is taken as evidence of nonlinearity. This measure is sensitive to the smoothness of the classifier's decision boundary as well as the overlap of the convex hulls of the classes.

If the complexity measures described above are based on Euclidean distance, this restricts them to situations in which all the features to be used in classification are numerical, as opposed to categorical, variables. Moreover, they are mostly based on a 1-Nearest Neighbor (1-NN) approach. Since each

reports a single complexity measure, they do not take account of the differential performance of classifiers across different classes, as is known to occur in situations where there is class imbalance. In the next section we propose and motivate some alternatives.

3.2.3 Complexity, Overlap and Imbalance

Let us assume the simple situation illustrated in Figure 3.1 where a single feature x is used to classify cases into Class 1 or Class 2. We assume perfect information in that the probability distributions of x conditional on class membership are known, say $f_1(x)$ in Class 1 and $f_2(x)$ in Class 2. In Figure 3.1, p_1 represents the portion of the probability density function of Class 1 for which $f_1(x) < f_2(x)$; this represents the probability that a case actually coming for Class 1 has an x value that makes it seem more likely to come from Class 2. Such cases would thus be expected to be misclassified even with perfect information on the conditional densities. Similarly, p_2 represents the probability that a case coming from Class 2 looks as if it really came from Class 1.

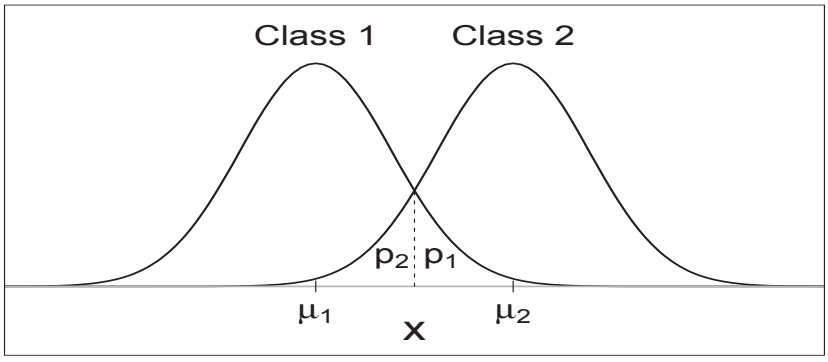

Figure 3.1: Overlap for two classes with a single feature, x

We measure the overlap of the probability density functions by $p_1 + p_2$. It will depend on the distance $\mu_2 - \mu_1$ separating the means of the conditional distributions relative to the standard deviations. In Figure 3.1 the distributions are assumed to be normal with equal standard deviations, so that

$p_1 = p_2$, and these probabilities are easily calculated given the means and standard deviations. We can also easily calculate the separation $\mid \mu_2 - \mu_1 \mid$ required to give prescribed values for p_1 and p_2. If instead we consider unequal standard deviations, multivariate features, or non-normal probability densities, it may not be possible to perform these calculations exactly. We can however, still assuming the densities are known, approximate p_1 by Monte Carlo integration, simulating a large number of cases from f_1 and calculating the proportion for which $f_1(x) < f_2(x)$; similarly for p_2.

Suppose now that a proportion π of cases in the population come from Class 2. Then the proportion of cases misclassified with perfect information will be $(1 - \pi)p_1$ from Class 1 and πp_2 from Class 2, thus a total of $(1 - \pi)p_1 + \pi p_2$; this represents the total classification error rate if we have perfect information and use likelihood to classify (unadjusted by prior information on class membership). Note that this depends on both the overlap and on the degree of class imbalance (unless $p_1 = p_2$).

In reality we do not have perfect information and training data must be substituted for knowledge of the conditional probability densities. If the classes are perfectly balanced ($\pi = 0.5$) then the likelihood criterion $f_1(x) < f_2(x)$ can be approximated by examining the k nearest neighbors of x for odd k and making a majority decision. However class imbalance distorts this test, tending to always favor the majority class.

3.2.4 Bayes Error

For a given dimension, the Bayes error rate can provide a lower bound on the error rate that can be achieved by any pattern classifier acting on that dimension [6, 7]. This rate is greater than zero whenever the class distributions overlap. When all the priors and class conditional likelihood are completely known, one can, in theory, obtain the Bayes error directly [7].

Let us assume a situation where a given pattern vector x needs to be classified into one of Y classes. Let $p(y_i)$ denote the prior class probabilities of class i, $1 \leq i \leq Y$, and $p(x|y_i)$ denote the likelihood i.e., the conditional probability density of x given that it is belongs to class i. The probability of

the pattern x belonging to specific class i, i.e., posterior probability $p(y_i|x)$ is given by the Bayes rule:

$$P(y_i|x) = \frac{p(x|y_i)p(y_i)}{p(x)} \tag{3.2}$$

where $p(x)$ is the probability density function of x and is given by:

$$p(x) = \sum_{i=1}^{Y} p(x|y_i)p(y_i) \tag{3.3}$$

The classifier that assigns a vector x to the class with the highest posterior is called the Bayes classifier. The error associated with this classifier is called Bayes error, which is expressed [7] as:

$$E_{Bayes} = 1 - \sum_{i=1}^{Y} \int_{Y_i} p(y_i)p(x|y_i)dx \tag{3.4}$$

where Y_i is the region where class i has the highest posterior. Obtaining the Bayes error from equation 3.4 requires evaluating the multi-dimensional integral of possibly unknown multivariate density functions. Since it is very difficult to estimate, the Bayes error can be computed directly only for very simple problem, e.g., problems involving Gaussian class densities with identical covariances. Alternatively, one can estimate the densities using general techniques (e.g., through Prazen windows [20]). Since this is not simple and error can be introduce during estimation, attention is focused on approximations and bounds for the Bayes error, which is either calculated through distributional parameters or estimated using training data .

3.2.4.1 Parametric Estimate of the Bayes Error

A simple bound for the Bayes error can be obtained from the Mahalanobis distance measure [6]. For the two class problem, let $\sum$ be a non singular, average covariance matrix ($\sum = p(y_1)\sum_1 + p(y_2)\sum_2$), and μ_i the class mean

vector, $i = 1, 2$. Then the Mahalanobis distance M given by:

$$M = (\mu_1 - \mu_2)^t \Sigma^{-1} (\mu_1 - \mu_2) \tag{3.5}$$

provides the following bound on the Bayes error [6].

$$E_{Bayes} \leq \frac{2p(y_1)p(y_2)}{1 + p(y_1)p(y_2)M} \tag{3.6}$$

The Mahalanobis distance is an easy and quick way of obtaining an approximation of Bayes error. However, it is not a particularly tight bound [21].

Another bound for a two class problem can be obtain from the Bhattacharyya distance given by [6]:

$$\rho = -\ln \int \sqrt{p(x|y_1)p(x|y_2)} dx \tag{3.7}$$

Assuming that the class densities are Gaussian, the Bhattacharya distance is given by [7]:

$$\rho = \frac{1}{8}(\mu_2 - \mu_1)^t \left(\frac{\sum_1 + \sum_2}{2} \right)^{-1} (\mu_1 - \mu_2) + \frac{1}{2} ln \frac{\left| \frac{\sum_1 + \sum_2}{2} \right|}{\sqrt{|\sum_1||\sum_2|}} \tag{3.8}$$

Using the Bhattacharyya distance the following bounds on the Bayes error can be obtain [6].

$$\frac{1}{2}(1 - \sqrt{1 - 4p(y_1)p(y_2)\exp(-2\rho)} \leq E_{Bayes} \leq \exp(-\rho)\sqrt{p(y_1)p(y_2)} \tag{3.9}$$

In general Bhattacharyya distance gives a tighter bound on the Bayes error than the Mahalanobis distance, but it has drawbacks. It requires knowledge of class densities and is more difficult to calculate. Even if the class distribution is known calculation for equation 3.7 is not generally practical. For more discussion see [22]. A tighter upper bound than Mahalanobis distance and Bhattacharyya distance based bounds is provided by Chernoff bound [7, 21]. Since Chernoff bound is computationally expensive and usually Bat-

tacharyya distance is preferred [5], we will restrict our computation of Bayes bound to Mahalanobis and Bhattacharyya based distances.

3.2.4.2 Non-Parametric Estimates of the Bayes Error

The computation of the bounds of two class problems presented in the last section depends on knowing (or estimating) certain class distribution parameters, such as priors, class means and covariance between classes. A method for estimating the Bayes error without requiring the class distributions of the data set is based on the nearest neighbor technique (NN). The Bayes error can be given in terms of the error of the NN classifiers. Given a two class problem with sufficiently large training data, the following results hold [23]:

$$\frac{1}{2}(1 - 2\sqrt{1 - 2E_{NN}}) \leq E_{Bayes} \leq E_{NN} \tag{3.10}$$

where E_{NN} is the error rate of nearest neighbors (majority class label of the k nearest neighbors is form other class). This result is independent of distance metric chosen.

3.3 Methodology: Data Complexity Measurement based on K-Nearest Neighbors

As stated earlier the Bayes error bound can be computed with the help of nearest neighbors (NN). We now extend it to the imbalance scenario. Note that in the binary class case equation 3.4 can be written as:

$$E_{Bayes} = P_1 \left(1 - \int_{Y_1} p(x|y_1)dx\right) + P_2 \left(1 - \int_{Y_2} p(x|y_2)dx\right) \tag{3.11}$$

where P_1 and P_2 are the prior probabilities for class 1 and class 2.

When two pattern classes are to be separated, errors occur in one of the two ways: either a pattern vector from class 1 is assigned to class 2 or

pattern vector from class 2 assigned to class 1. Let P_i be the prior probability of class y_i and $p(X|y_i)$ be the conditional probability density of pattern vector X when it comes from class y_i. Applying the pattern vectors to an optimum Bayes classifier, the first error rate can be written as:

$$E_1 = \text{Prob}\left\{P(Y_1)p(X|y_1) < P(Y_2)p(X|y_2) \mid X \in y_1\right\} \tag{3.12}$$

and the second error rate as:

$$E_2 = \text{Prob}\left\{P(Y_1)p(X|y_1) > P(Y_2)p(X|y_2) \mid X \in y_2\right\} \tag{3.13}$$

The total error rate is then:

$$E^* = E_1 + E_2 \tag{3.14}$$

Our approach focuses on the local information for each data point via nearest neighbors, and uses this information to capture data complexity. Calculation involves finding the k nearest neighbors of every data point in the class, where k is odd. If the majority of the neighbors are in the same class, this point is designated as easy to classify; if most of the neighbors are in the opposing class, it is difficult. Complexity can be measured as follows:

For a single point (case j) in class Y_j with neighborhood N_j

$$CM_{k(j)} = \text{I}\left(\frac{\text{number of patterns } j\prime \text{ in } N_j \text{ with } Y_j\prime = Y_j}{k} \leq 0.5\right) \tag{3.15}$$

where I(.) is the indicator function. The overall measure is

$$CM_k = \frac{1}{n}\sum_{j=1}^{n} CM_{k(j)} \tag{3.16}$$

The overall complexity measure is the proportion of points classified as difficult, but this can be decomposed into separate measures for the complexity of each class corresponding to decomposition in equation 3.11 and 3.14.

$$CM_{k,i} = \frac{1}{n_i} \sum_{Y_j=i} CM_{k(j)} \tag{3.17}$$

Typically k should be large enough to use this approach properly (i.e., we can seen the trend in error rate for the various choices of k). We recommend using an odd number for k to avoid ties. Deciding on k is a non trivial problem in the literature. We suggest:

$$k = \underset{k'}{\operatorname{argmin}}(CM_{k'} - CM_{k'-2}) \leq \alpha \tag{3.18}$$

where α is a chosen threshold value. Our experiments suggest that low overlap needs only a low value of k, whereas for higher overlap the complexity measure only stabilizes at higher k. Solving for k in equation 3.18 analytically is not possible so we use a data driven approach, increasing k until CM does not increase appreciably (see Figure 3.2 for visualization). This formulation will be materialized specifically in section 3.5.

3.3.1 Distance metric

The identification of nearest neighbors presupposes a distance metric that uses the available features or variables to measure the dissimilarity between cases. A complimentary concept is proximity, that measures the degree to which a pair of cases are similar. A proximity measure can easily be converted into a distance, and vice versa. In our study we examine both Euclidean distances and proximity measured by the Random Forest [24] technique to find nearest neighbors. We now discuss their relative merits for our purpose from a theoretical perspective.

Euclidean Distance: This is widely used and intuitively appealing. In two or three dimensions, it is easily visualized as the length of the straight line connecting two points x and y. In p dimension the Euclidian distance is defined as:

$$d(x, y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2} \tag{3.19}$$

where $x = (x_1, \ldots, x_p)$ and $y = (y_1, \ldots, y_p)$.

One problem with this method is that Euclidean distance is sensitive to large values: in other words, it is sensitive to outliers. Secondly this method is scale dependent: if we multiplied all the values of one feature by a constant, for example changing height from meters to centimeters, this would change its contribution to the overall distance and possibly affect the set of nearest neighbors. Thirdly it is not suitable for data with categorical features since it requires all contributing variables to be numerical.

Another area needing extra attention is that when the scales of original variables are widely different, it is commonly recommended to standardized the original variables first, before applying Euclidean distances, in order to alleviate the adverse effect of different scales associated with the original variables. We use standardization of variables for all data sets for this chapter and throughout the thesis, before applying Euclidean distances.

Random Forest: A Random Forest [24] is a collection of classification trees generated randomly using a two-step process. First a bootstrap sample of cases is taken from the training data set. Secondly, a classification tree is grown in which features are selected from a randomly chosen subset at each node. It is used as a classifier by combining the individual classifications from each tree in the "forest".

This process can be used to calculate a proximity measure as follows. The original data are labeled as class 1, and a synthetic data set is constructed by simulation and labeled as class 2 . We then try to classify the combined data with a Random Forest. There are two ways to simulate the class 2 data:

1. The class 2 data are sampled from the product of the marginal distributions of the variables (by independent bootstrapping of each variable).

2. The class 2 data are sampled uniformly from the hypercube containing the data (by sampling uniformly within the range of each feature).

In the Random Forest algorithm, Breiman [24] constructs similarity matrix as follows:

1. Send all learning examples down each tree in the forest

2. If two examples land in the same leaf, increment corresponding element in similarity matrix by 1

3. Normalize the matrix with the number of trees.

Breiman [24] stated that the proximities between cases n and k form a matrix $\{prox(n,k)\}$. From the definition, it is easy to show that this matrix is symmetric, positive definite and bounded above by 1, with the diagonal elements equal to 1. It follows that the values $1 - prox(n,k)$ are squared distances in a Euclidean space of dimension not greater than the number of cases.

The idea here is that the real (i.e., 'class 1') data points that are similar to one another will frequently end up in the same terminal node of a tree. The proximity matrix gives, for each pair of cases, the proportion of trees in the forest for which the two cases are in the same terminal node. This can be converted to a dissimilarity matrix by subtracting all the elements from 1. Liaw and Wiener [25] showed how clustering or multi-dimensional scaling using this dissimilarity measure can be used to divide the original data points into groups for visual exploration. We use random forest proximity measure for its many theoretical advantages (for more detail see Shi and Horvath, [26]).

3.3.2 Visualization

For a given distance metric, the corresponding dissimilarity matrix can be used to represent the data in low-dimensional space. For this we use Multidimensional Scaling (MDS), a statistical technique originating in psychometrics [27]. The data used for MDS are dissimilarities between pairs of objects. The main objective of MDS is to represent these dissimilarities as distances between points in a low dimensional space such that the distances correspond as closely as possible to the dissimilarities. This representation can then be displayed visually using graphical techniques.

Visualization of data based on distances is a powerful methodology ("a picture is worth a thousand words"). It gives excellent training in visual

thinking/cognition "...us(ing) not language but mental graphics system, with operations that rotate, scan, zoom, pan, displace, fill-in ..." (Pinker, 1994, p73) [28]. Visualization is primarily dependent on the analogy between similarity and proximity (and hence between dissimilarity and distance) and can be enhanced by movement/interaction and color.

In our situation, visualization can be used in conjunction with our complexity measure to see the degree of overlap between two class. It can help us to analyze the performance of classifiers on a given data set, and reveal structures in the data that affect this performance.

3.4 Simulation Study

In our simulation study we want to investigate the following:

- Quantification of complexity based on k-nearest neighbor for various choices of k;
- Effect of degree of overlap and class imbalance on complexity;
- Comparison with existing techniques for data complexity;
- Comparison of Random Forest distance with Euclidean distance.

3.4.1 Design

Simulation data were generated from two p-dimensional multivariate normal distributions with equal covariance matrices Σ and with $\mu_0 = (0, 0, ..., 0)'$ and $\mu_1 = (m, m, ..., m)'$. Scenarios are defined by Δ_μ the vector of mean differences for the classes. We varied the class size n_1 and n_2, where $n_1 + n_2 = N$, to give different imbalance scenarios. (However, this generalizes by a change of co-ordinates to any system with equal covariance and different means.) For a given Δ_μ and equal sample sizes i.e., $n_1 = n_2$, the optimal (i.e. Bayes) error rate is:

$$E_{Bayes} = 1 - \Phi\left(\frac{1}{2}\Delta_\mu \times \sqrt{p}\right) \tag{3.20}$$

Table 3.2: k-Nearest Neighbor Simulation Design

Bayes Error	Δ_μ	n_1	n_2
0.4	0.29	1000	1000, 500, 300, 200, 90
0.3	0.61	1000	1000, 500, 300, 200, 90
0.1	1.48	1000	1000, 500, 300, 200, 90

where Φ is the CDF of the standard normal distribution. In general the estimated error rates should be interpreted as optimistic because the analysis uses appropriate models for the data; for real data one does not know what classification methods are appropriate. In case of imbalance equation 3.20 becomes:

$$E_{Bayes} = 1 - \Phi\left(\frac{\log(P_1) - \log(P_2)}{4 \times \Delta_\mu} + \frac{1}{2}\Delta_\mu \times \sqrt{p}\right) \tag{3.21}$$

where P_1 and P_2 are the prior probabilities for class 1 and class 2.

Table 3.2 shows the values of Bayes error, n_1 and n_2 chosen for the experiments. We verified the Bayes error using Monte Carlo integration by equation 3.12 and 3.13.
For dimensions p: 2, 5 and 10, multivariate normal distributions were also simulated and analyzed, but due to similar nature of pattern/results, these are not presented.

All programs were written in the statistical environment R [29] and relevant R function are given in Appendix 7.2. For the construction of tree models we used the `RPART` package [30], for nearest neighbor based on Euclidean distance we used R package `class` [31], with a little modification of the input and output and to compute Random Forest distances we used the R package `randomForest` [24].

3.4.2 Results

Detailed results for $p = 3$ are tabulated in Appendix 7.2. Here we discuss our main findings and give illustrative graphical summaries. One finding of the study is that the complexity measures based on N1 tend to over-

estimate the theoretical overlap, whereas complexity measures based on k-NN where $k > 2$, like 5NN and 7NN, improve estimation of the overlap. A typical example for $p = 3$ is shown in Figure 3.2 for the balanced case ($n_1 = n_2 = 1000$, top left). The complexity measure is shown first for Class 1, then for Class 2, for 3-, 5-, 7-and 9-NN, with error bars to show variability across simulations. B1, B2 give the Bayes error, i.e., knowing the probability distributions of the two classes. N1 represents the complexity measurement proposed by the Ho and Basu [12] but here decomposed into separate measure for each class. From Figure 3.2 we can see that, if we have high overlap for the balanced distribution, complexity is the same for both classes, and as we increase the number of neighbors it becomes more compatible with actual degree of overlap (B1 and B2). When we reduce the degree of overlap, complexity measurement for both classes decreases as expected, and the results for different values of k become more similar. This is also true for the complexity measurement N1: with less overlap we get lower N1 values for both the classes although it is overestimating the Bayes error. Another complexity measure proposed by Ho and Basu [12] N2 gives 0.98 ± 0.01 for all degrees of overlap, so is insensitive to the degree of overlap.

We can see from Figure 3.2 how complexity of the minority class becomes high, in comparison to the majority class, when class imbalance increases in the presence of overlap. The class overlap is shown by the corresponding Bayes error of respective classes. For the severe imbalance data sets (minority class under 10%, Figure 3.2, bottom right), the Bayes error is near zero for class 1, when there is severe class imbalance and high overlap. When this is compared to our proposed complexity measure, its complexity become zero when k increases. This corresponds to the domination of the majority class when we classify severely imbalanced data sets. Similarly for the other Bayes error (less overlap) scenario (see the Figure 3.2, top right and bottom left), we can find relatively less complexity for minority class and more complexity for the majority class. This indicates that class imbalance with high overlap is fatal for minority class accuracy. When comparing to the existing complexity measurements (proposed in the literature by Ho and Basu [12]) we find that these complexity measurement are insensitive to class imbalance in

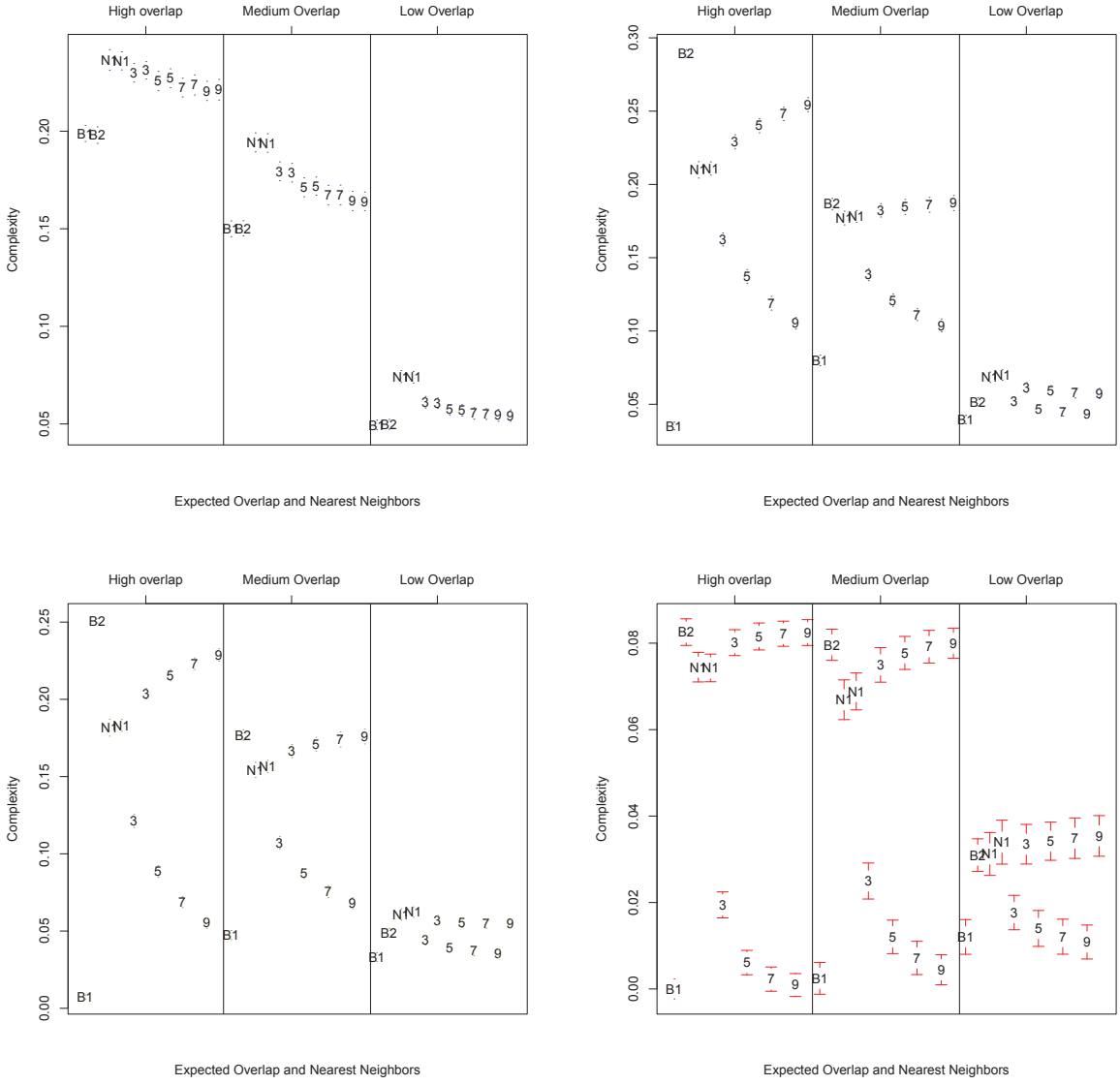

Figure 3.2: Complexity measurement for simulated 3 dimensional normal multivariate distributions. The different panels show different degree of imbalance from the balanced distribution 1000 observation is each class(top left), imbalanced data set 1000 observation in class 1 and 500 in class 2 (top right), imbalanced data set 1000 observation in class 1 and 300 in class 2 (bottom left), to severely imbalanced 1000 observation in class 1 and 90 in class 2 (bottom right). The complexity measure is shown first for Class 1, then for Class 2, for 3-, 5-, 7- and 9-NN with error bars to show variability across simulations. B1, B2 give Bayes error and N1 gives fraction of points on class boundary.

overlapped cases, i.e., N1 give the same complexity for both of the classes even though in reality complexity of minority class should be much higher than the majority class. This measure is sensitive to class overlap, but not class imbalance. However, N2 gives us complexity of 0.67 ± 0.02 in all the overlapped cases, so this measure is insensitive to class overlap. Both N1 and N2 depend on the total number of observations but not the class sizes, making them unsuitable in class imbalance situations.

When we applied Random Forest distances from balanced to severe imbalanced data sets we obtained results similar to those for Euclidean distance. We also did further simulations, not shown here, using the Random Forest based complexity measure on mixed type data sets (some continuous and some categorical features) that gave broadly similar results.

To summarize the findings of this simulation study:

1. k-nn where $k > 2$ may be used in order to capture the complexity of classes with imbalanced data;

2. The complexity measures proposed by the Ho and Basu [12] work well for balanced data but cease to capture complexity in imbalanced data sets. N1 and N2 are insensitive to class imbalance, which is a disadvantage as severe imbalance (minority class $\leq 10\%$) seriously degrades classifier performance;

3. As k increases, k-nn becomes more consistent with the Bayes error;

4. As the imbalance increases our complexity measure for the minority class increases appropriately.

The main purpose of our complexity measure is to see the effect of class imbalance with different levels of complexity. As N1 is insensitive to class imbalance, so we have to consider a higher nearest neighbor approach to measure actual complexity for each class.

3.5 Real Data Sets

3.5.0.1 Threshold to regulate k in Complexity Measurement (CM)

The choice of number of nearest neighbors, k, can be informed by the finding of the simulation study. For low overlap we prefer lower k, because increasing the nearest neighbor gives the same result. For higher overlap the complexity measure will stabilize at higher k, so we use the k where the complexity for the minority class does not change appreciably with further increase. For example Fig 3.2, we can see there is not much difference between the complexity measure by k=7 and k=9, so we prefer k=7. That is why we put the condition as define in equation 3.18. We fix the threshold as:

$$\alpha = 0.025 \times \text{number of observations in minority class} \tag{3.22}$$

We keep increasing k until the complexity for a specific data set does not change by more than α; then k satisfies equation 3.18. We choose 0.025 as a rule of thumb, which we have found appropriate for the data sets we are using, but of course it can be changed to a more appropriate value, if we have some prior knowledge about the data sets, or if the misclassification cost is known. We make it adaptive for different imbalanced data sets by including the number of observations of minority class, i.e., for relative high minority class size α will be high, and for severe imbalances cases α will be low.

To investigate the performance of our complexity measure with real data, we have used 20 data sets from the UCI data repository [32]. Since there is only a limited number of binary class problems available, we have transformed multi-class data sets using a 1-vs-others approach. By using different choices for class 1, the same data set can be used to create different degrees of imbalance and complexity. Information about these data sets is summarized in Table 3.3, where Size is the total number of observations, Attributes is the number of dimension, Target is the minority class while the majority class is the union of all other classes. $\sharp Min$ and $\sharp Maj$ are the sizes of the minority and majority class, and $\%Min$ shows the proportional size of the minority

Table 3.3: Description of UCI data sets

Data sets	Size	*Attributes*(p)	Target	$\sharp Min$	$\sharp Max$	$\%Min$
Abalone1	4177	8	Ring=7	391	3786	9.4
Abalone2	4177	8	Ring<7	448	3729	10.7
Abalone3	4177	8	Ring=19	32	4145	0.77
Balance	625	4	Balance	49	576	7.8
Car	1728	6	acc	384	1344	22.2
Cmc	1473	9	class 2	333	1140	22.6
Haberman	306	3	class 2	81	225	26.5
Ionosphere	351	34	bad	126	225	35.9
Letter	20000	17	A	789	19211	3.9
Pima	768	8	class 1	268	500	34.9
Satimage	6435	36	class 4	626	5809	9.7
Vehicle1	846	19	opel	212	634	25.1
Vehicle2	846	19	saab	217	629	25.7
Vehicle3	846	19	bus	218	628	25.8
WDBC	569	34	malignant	212	357	37.3
WPBC	198	34	recur	47	151	23.7
Yeast1	1484	8	nuc	429	1055	28.9
Yeast2	1484	8	ME3	163	1321	11
Yeast3	1484	8	exc	35	1449	2.4
Yeast4	1484	8	ME1	44	1440	3.0

class. In order to preserve the class ratio, stratified 5-fold cross validation was used to obtain training and test subsets (ratio 4:1) for each data set.

In order to produce unbiased results we used Classification Tree (CT) as a base learner. We applied CT to each data set and measured the learning performance on the smaller class by the sensitivity (accuracy of minority class) and G-Mean. Table 3.4 illustrates the results using the sensitivity (accuracy of minority class) and G-Mean along with complexity proposed by Ho and Basu (F1, L3, N1) [12] and our proposed complexity measurement (CM). Sensitivity is chosen because we are interested in accuracy for the minority class rather than overall accuracy. We evaluated the algorithms using the metric G-Mean defined as $\sqrt{TP \times TN}$ [33], which corresponds to the geometric mean between the correct classification rates for positive (sensitivity) and negative (specificity) examples, respectively. All analyses were done with 5-fold cross validation with the resulting sensitivity values averaged to a single score. We use these scores as the general indicator of whether a data set is easy or difficult to learn. We want to see how well the

Table 3.4: This tables compares Sensitivity values and G Mean on UCI Imbalanced data sets. Mean values for each data set were calculated for 5 runs with different test subsets obtain from stratified 5 fold cross validation The first column lists the data sets used. The following columns(2-3) shows the ratio of minority class in the data set, columns (4-6) the complexity measure used in the literature and column(7) our proposed complexity measure.

Data Set	% Min	IR	F1	L3	N1	CM	Sensitivity	G-Mean
WDBC	37.258	1.684	3.568	0.007	0.831	0.080	0.880	0.910
Ionosphere	35.897	1.786	0.609	0.980	1.129	0.405	0.620	0.870
Pima	34.896	1.866	0.576	0.500	1.008	0.427	0.620	0.680
Yeast1	28.908	2.459	0.242	0.500	0.879	0.564	0.480	0.629
Haberman	26.471	2.778	0.185	0.0.497	1.024	0.346	0.390	0.540
Vehicle3	25.768	2.881	0.169	0.368	0.049	0.037	0.944	0.951
Vehicle2	25.650	2.899	0.381	0.0.231	0.284	0.452	0.456	0.623
Vehicle1	25.059	2.991	0.186	0.351	0.936	0.566	0.560	0.690
WPBC	23.737	3.213	0.142	0.780	0.931	0.745	0.440	0.580
Cmc	22.607	3.423	0.245	0.500	0.908	0.718	0.290	0.500
Car	22.222	3.500	0.749	0.380	0.940	0.290	0.840	0.890
Yeast2	10.984	8.104	2.751	0.5	0.679	0.325	0.720	0.830
Abalone2	10.725	8.324	2.947	0.520	0.365	0.489	0.590	0.750
Satimage	9.728	9.280	0.375	0.500	0.556	0.345	0.570	0.740
Abalone1	9.361	9.683	0.879	0.650	0.489	0.903	0.200	0.430
Balance	7.840	11.755	0.001	0.500	0.971	1.000	0.000	0.000
Letter	3.945	24.349	17.518	0.980	0.686	0.000	0.910	0.950
Yeast3	2.965	32.727	4.198	0.500	0.122	0.273	0.625	0.781
Yeast3	2.358	41.400	2.302	0.500	0.594	0.486	0.400	0.612
Abalone3	0.766	129.531	0.530	0.500	0.731	1.000	0.100	0.196

complexity measures correlate with this sensitivity measure.

Our simulation experiment suggested that using k-NN instead of 1-NN gives better behavior, moreover choosing $k > 1$ eliminates the influence of outliers [34], so we have used k-NN for our complexity measure of real data sets.

Table 3.4 is organized by Imbalance Ratio (IR), defined as the number of negative class examples divided by the positive class examples.

We examined the relationship between our complexity measure for the minority class and the sensitivity of classification with respect to the minority class. This relationship between data complexity measures and learner

performance for the minority class is illustrated in Figure 3.3. From Figure 3.3 (top left), it is clear that there is a strong linear relationship between our CM and sensitivity value, but a weak or low relationship among other complexity measure with sensitivity value. The resulting correlation of data complexity and learning performance was found to be -0.93 for sensitivity values and -0.90 for G-Mean. In contrast the correlation analysis for the other complexity measures is very low: Fisher Discriminant ratio (F1) and sensitivity or G-Mean have a correlation of 0.43, which is the strongest among the proposed complexity measures of [12]. There is very little or no correlation found using L3 or N1.

We employed MDS to visualize the data and to demonstrate the geometrical significance of our complexity measure. The degree of correspondence between the distances among points implied by MDS map and the matrix input by the user is measured (inversely) by a stress function. Here we measured the stress of the MDS plot, not for the sake of dimension choice (i.e., how many dimension will be appropriate for representation), but to compare the stress functions of the two different distance metrics. In our case these are Euclidean and Random Forest distances. The smaller the stress, the better the representation. However, the problem with stress is that with increasing dimensions, you must estimate an increasing number of parameters to obtain a decreasing improvement in stress. The result is a model of the data that is nearly as complex as the data itself. In other words for high dimension we may end up in measuring a high stress value.

We discuss results for two of the data sets in detail (one easy and one hard).

The Winconsin Diagnostic Breast Cancer (WDBC) from the UCI Machine Learning Repository has 32 variables computed from a digitized image describing the characteristics of the cell nuclei present in each of 569 images. The class variable of interest here is the diagnosis, either benign (majority class) or malignant (minority), with particular emphasis on the detection of the minority class. Excluding patient ID and diagnosis (class label) columns, we used 30 features for our analysis. All variables in the data sets standardized before measuring Euclidean distances.

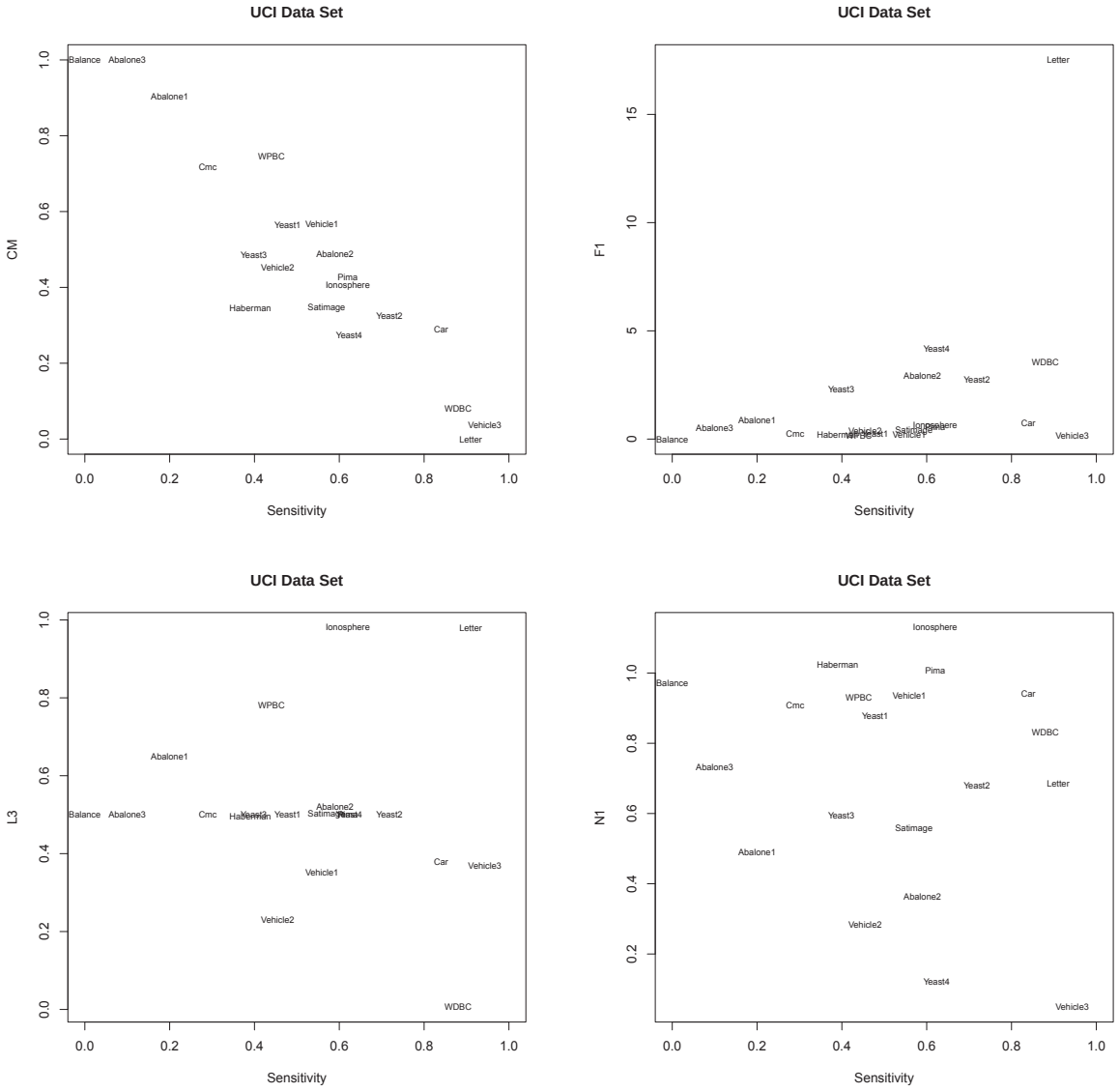

Figure 3.3: Scatter plot for Sensitivity values and Complexity Measurement for UCI data set, top left is our proposed Complexity Measurement (CM), top right is Fisher Discriminant Ratio (F1), bottom left is Nonlinearity of nearest neighbor or linear classifier (L3) and bottom right shows Fractions of points on class boundary (N1). Every data set is represented by its name

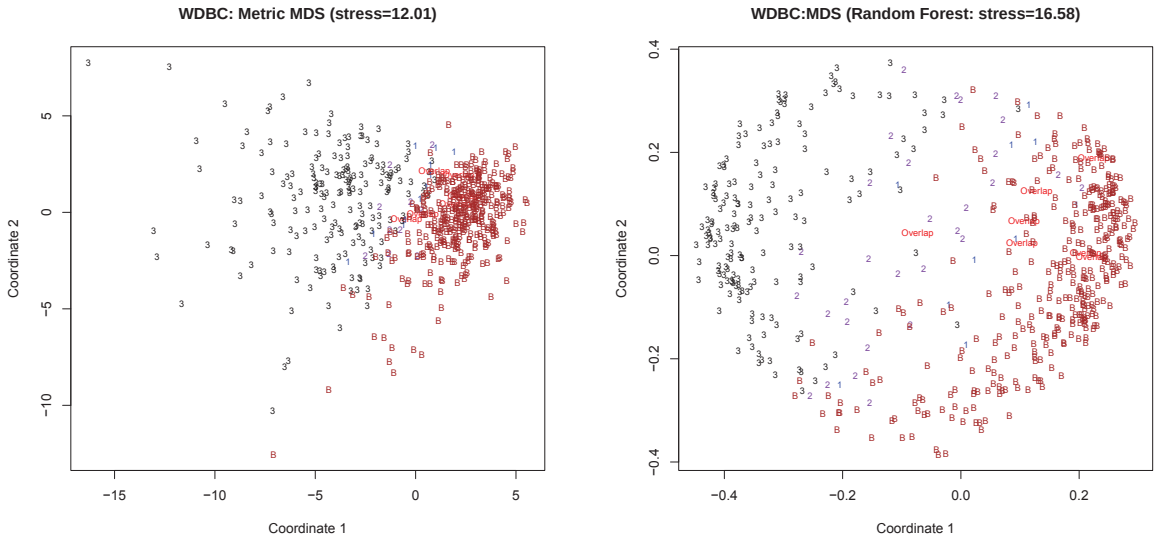

Figure 3.4: Visualization by MDS of Euclidean distance (left) and Random forest distances (right) for UCI breast cancer dataset. The symbols and color combination shows two different classes: B (Maroon) benign cases and malignant cases (minority class) shown by the number of 3-nearest-neighbours (Black=3, Purple=2, Blue=1 and Red=Overlapped)

Figure 3.4 plots each case by 2-d MDS using the class variable as the plotting symbol and the number of 3-nearest neighbors belonging to the minority class. Figure 3.4 shows two separate clusters. When we apply our complexity technique using 3NN method (k determined by equation 3.18), we can classify each case of the minority class according to how many of its neighbors are of the same class. Thus those points not on the boundary are represented by 3, points nearest to the decision boundary are represented by 2, 1 or 0 (Overlap), where 2 is more tilted towards its own class, 1 and 0 points are more tilted to other class, these being points more vulnerable to misclassification. Cases scoring zero are denoted "Overlap" as they are surrounded by cases from the opposing class. Note that this method (CM) is completely general and can be applied to any distance function. Figure 3.4 suggests that there is not much overlap between the classes, so this data set can be labeled as easy, in that we expect most classifiers to perform well in this scenario. Indeed, Table 3.4 shows that our classifier gives reasonable

accuracy for the minority class with sensitivity of 88%. We measure the stress function to show how well the (first 2 dimension) MDS plot represents the data. A stress value of stress= 12.01, seems to be high, but this is expected, as this is a high dimensional data set, and the stress function is highly dependent on number of dimensions. Although this stress function is high, we can visualize that these two dimension are appropriate for representation, as we can see two separate cluster of classes and our complexity measurement is also estimating a low value. Recall that our CM measures the complexity of the original 30- dimensional data set.

When we use Random Forest distance in place of Euclidean distance for the WDBC data set, we get a similar conclusion, but the visualization, shown in the right panel of Figure 3.4, is dramatically different. Random Forest based MDS shows the structure of the data in a very different light, with the cases more separated in space and a clearer view of the boundary, that runs diagonally across the right hand side. This is indicated by our complexity measure taking the values 2, 1 and overlap. The stress is 16.58 for the Random Forest distances, which is higher than the Euclidean distance stress measurement. However we can see a similar pattern in both MDS (Euclidean and Random Forest distances) plots.

Next we consider a more severely imbalanced data set, Abalone (Age=7). We can determine precisely the age of an abalone by cutting the shell through the cone, staining it and counting the number of rings through a microscope. However this task is cumbersome and time consuming. Thus we want to predict the age of an abalone from physical measurements which are easier to obtain. These measurements include sex and seven other features such as length, diameter and height. The original response variable of this data sets is the age of abalone, ranging from one to 29, which we use to divide the whole data set into two classes. We use age=7 for the minority class, resulting in 391 (9.4%) cases out of 4177 samples in the minority class.

If we visualize the data set using Euclidean MDS as in Figure 3.5, there is no visible separation between the two classes. One explanation may be that these two dimensions may not be appropriate to show the separation between the classes or to visualize the decision boundary. But the stress is

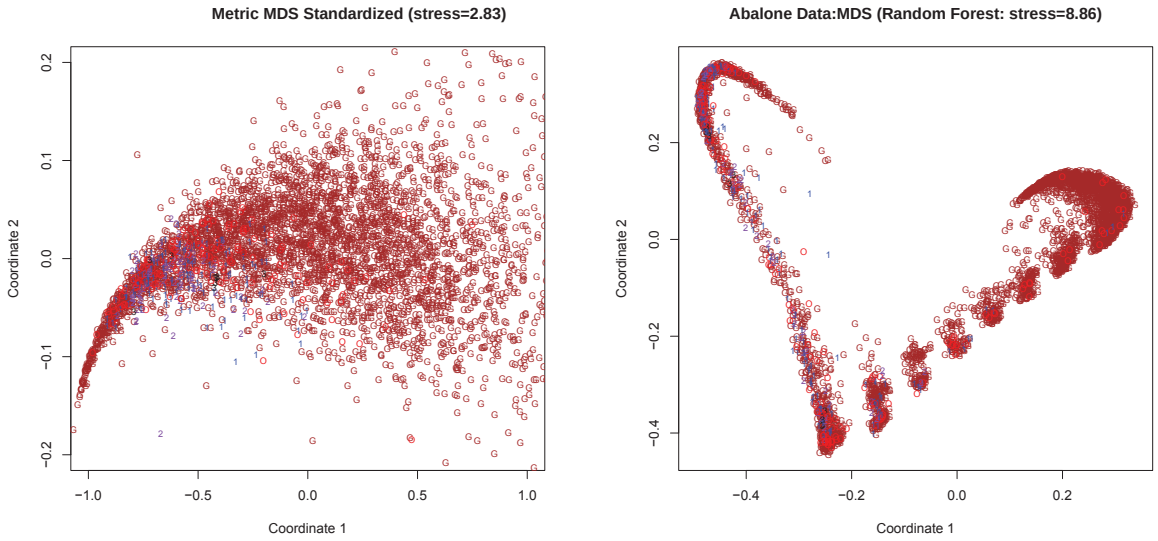

Figure 3.5: Visualization by MDS of Euclidean distance (left) and Random forest distances (right) UCI abalone dataset.. The symbols and color combination shows two different classes:G (Red) shows the majority class, where as minority class (Age=7) shown by the number of 3-nearest-neighbours (Black=3, Purple=2, Blue=1 and Red=O)

2.83 for this 2-d MDS plot, which is much lower than stress in the WDBC data sets. This lower stress value is due to lower number of dimensions which in this case is 8. If we apply our complexity measure on the abalone data set, as expected we have a lot of overlap for the minority class, as shown by a lot of "O" (red) and "1" (blue) which represent respectively those data points of the minority class being surrounded by the majority class with only zero or one out of three belonging to the same class respectively. Hence these data points are more vulnerable to misclassification. Since this data set has more overlap, which is evident from Figure 3.5 and from our complexity measure, we are not expecting much classification accuracy for the minority class, which can be validated from Table 3.4.

Using the Random Forest distance gave us the same overall picture, i.e., high overlap between the minority and majority class, stress is 8.86 which is higher than the Euclidean distance. Visualization using Random Forest distance, shown in Figure 3.5 (right panel), reveals some unexpected structure

in the data that requires further investigation.

3.6 Discussion and Future Work

In this chapter we have further explored work done by Weng & Poon [11] and extended work done by Ho and Basu [12] in data complexity measurement, specifically for the scenario of imbalance. In this chapter our focus is to explore the connection between the imbalanced data sets problem and data complexity. Therefore we have focused more on tools for quantification of the problem associated with the imbalance data sets rather than solving the problem, which we see as the first step for answering questions such as:

1. What is the nature of the class imbalance problem, i.e. in what situations and to what extent does class imbalance hinder the performance of standard classifiers.

2. What are the possible solution in dealing with class imbalance problems, and how well do they perform? For example, when should we employ over-sampling or under-sampling instead of fixing a class distribution or size.

Our simulation study showed that in cases of imbalance our complexity measurement is within the proposed bounds of Bayes error given in literature, such as Mahalanobis bounds, Battacharayya distance [7] (see Table 3.5). It is close to the lower bound in the case of class imbalance and almost lower bound in cases of severe imbalance. Although the nearest neighbor technique is used in the literature [23] to estimate Bayes error, our contribution is to extend it to class imbalance situations and using the Random Forest distance matrix for categorical and mixed type data sets.

Unlike previous authors, we calculate a complexity measure for each class, being the proportion of difficult cases belonging to that class, where 'difficult' means having a majority of k nearest neighbors in the opposing class. This idea could perhaps be extended by weighting each case according to the

proportion of neighbors in the other class, rather than making a majority decision.

We have shown that our complexity measure is able to capture a reasonable amount of data complexity despite the diverse nature of data sets. We used simulated data sets to build up a complexity model and to describe different types of imbalance data sets. Using MDS enables us to visualize the data sets and to visually interpret some of the findings of our complexity measures. From the existing data complexity measures proposed by Ho and Basu [12] only Fisher Discriminant analysis ratio (F1) showed some correlation, which is interesting because it has been found useful studies of prototype selection [35] and it is showing some correlation in our analysis in the imbalance framework as well.

By basing our measure on k-NN where $k > 2$ we are able to identify the points near the decision boundary. This information could be useful in interactively changing some aspect of a data set, or a parameter of a learning algorithm. For example if we select points near to the decision boundary, we can see what effect it can produce on the decision boundary if we over-sample these data points by SMOTE [36] or any other over-sampling technique, or under-sample the majority class near to these data points. We can then see how the decision boundary changes for various classifiers and explore its effect on accuracy for the minority class.

Another advantages of using k-NN with $k > 2$ is to extend the idea of COG (classification using local clustering) [37] using our complexity measure. They cluster large classes into small clusters using a standard clustering algorithm and then classify each cluster against the minority class. The motivation behind their clustering idea is to get more balance and to divide the complex concepts into simpler ones (i.e., nonlinear decision boundaries to linear separable boundaries). This may not be appealing when there is a high degree of overlap between the class distributions, since their methodology will result in large misclassification error in either of the classes, and we may get better results for the minority at the expenses of majority class. Using the complexity concept we could divide the space into more manageable parts by dividing into different clusters based on nearest neighbor misclassification; for

example, our 3NN approach will give us 4 clusters based on how many nearest neighbors of the minority class cases are in the majority class (0, 1 ,2 or 3). Thus we could separate difficult clusters from easy ones, so that the learning process is more flexible and adaptive. For example over-sampling or under-sampling can be done more purposefully instead of randomly. Moreover by doing so we can capture intra vs inter class imbalance and small disjunct as discussed by Jo [2]. The importance of different levels of complexity can be found in the study of Lim and Sohn [38], where they used cluster based approach (k mean algorithm) to define each cluster likely to have a different level of complexity, for their case study to classify a borrowers credibility.

In addition we hope that our complexity measure will help us to decide which classifier should be used for a specific data set. We have notice that if there is high overlap, a linear classifier appears to produce much better results than nonlinear; however this needs further extensive investigation. Under-sampling or over-sampling based on our complexity measure seems to be a promising approach and will be investigated in Chapters 4 and 5. The effect of dimension reduction on the complexity and on the accuracy of the minority class is another direction and will be investigated in Chapter 6.

References

[1] Devroye L., Any discrimination rule can have an arbitrary bad probability of error for finite sample size. IEEE Trans Pattern Analysis and Machine Intelligence, Vol. 4, no. 2, pp. 154-157, 1982.

[2] Jo, T., Japkowicz, N., Class imbalances versus small disjuncts, SIGKDD Explorations, Vol. 6, pp. 429-450, 2004.

[3] Provost, F., and Fawcett, T., Robust classification for imprecise environments, Machine Learning Journal, Vol. 42, pp. 203-231, 2001.

[4] Prati, R. C., Batista, G. E. A. P. A., and Monard, M. C., Class imbalances versus class overlapping: an analysis of learning system behavior. In MICAI, pp. 312-321, 2004.

[5] Fukunga, K., and Krile, T. F., Calculation of Bayes Recognition Error for Two Multivariate Gaussian Distributions, IEEE Transaction on Computer, Vol. 18, no. 3, pp. 220-229, 1969.

[6] Devijer, P. A., and Kittler, J., Pattern Recongnition: A Statistical Approach. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[7] Fukunga, K., Introduction to Statistical Pattern Recognition, 2nd edition. Boston: Academic Press, 1990.

[8] Bock, H.-H. (ed.), Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information From Complex Data. Berlin, Heidelberg: Springer, 1999.

[9] Tumer, K., and Ghosh, J., Bayes Error Rate Estimation Using Classifier Ensembles, Smart Engineering System Design, 5, 2003, 95-109.

[10] Ho, T. K., A Data Complexity Analysis of Comparative Advantages of Decision Forest Constructors, In Pattern Analysis and Application, pp. 102-112, 2002.

[11] Weng, C. G. and Poon, J., A data complexity anylysis on imbalanced datasets and an alternative imbalance recovering strategy, Proceeding of the IEEE/WIC/ACM International Conference on Web Intelligent, 2006.

[12] Ho, T. K., Basu. M, Complexity measures of a supervised classification problems, IEEE Trans Pattern Analysis, Machine learning, Vol. 24, no. 3, pp. 289-300, 2002.

[13] Bernado-Mansilla, E., Ho, T., K., Domain of competence of XCS classifier system in complexity measurement space. IEEE Trans Evol Computation, Vol. 9, no. 1, 82-104, 2005.

[14] Li, Y., Member, S., Dong, M., Kothari, R., Classificability-based omnivariate decision trees, IEEE Trans Neural Netwrok Vol. 16, no. 6, pp. 1547-1560, 2005.

[15] Baumgartner, R., Somorjai, R., L., Data complexity assesment in undersampled classification of high dimensional biomedical data, Pattern Recognition Letter, Vol. 12, 1383-1389, 2006.

[16] Garcia, V., Mollineda, R., Sanchez, J., S., On the k-NN performance in challenging scenario of imbalance and overlapping. Pattern Analysis Application, Vol. 11, no. 3-4, pp. 269-280, 2008.

[17] Sanchez, J., Mollineda, R., and Sotoca, J., An analysis of how training data complexity affects the nearest neighbor classifiers. Pattern Analysis Application. Vol. 10, no. 3, 189-201, 2007.

[18] Friedman J. H., Rafsky L. C., Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests. Annals of Statistics, Vol. 7, no. 4, pp. 697-717, 1979.

[19] Hoekstra A, and Duin R. P. W., On the nonlinearity of pattern classifiers. Proceedings 13th ICPR, Vienna, pp. 271-275, 1996.

[20] Parzen, E., On the estimation of a probability density function and the mode. Annals of Math. Stats.,Vol. 33, pp. 1065-1076, 1962.

[21] Duda, R. O., Hart, P. E., and Stork, D. G., Pattern Classification, 2nd Edition, New York, NY: John Wiley and Sons.

[22] Djouadi, A., Snorrason, O. and Garber, F. D., The quality of training sample estimates of the Bhattacharyya coefficient. IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 12, no. 1, pp. 92-97, 1990.

[23] Cover, T. M., and Hart P. E., Nearest Neighbor pattern classification. IEEE Transactions on Information Theory, Vol. 13, pp. 21-27, 1967.

[24] Breiman, L., Random Forest, Machine Learning, Vol. 45, no. 1, pp. 5-32, 2001.

[25] Liaw, A. and Wiener, M., Classification and regression by Random Forest. R News. Vol. 2, pp. 18-22, 2002.

[26] Shi, T. and Horvath, S., Unsupervised Learning with Random Forest Predictors. Journal of Computational and Graphical Statistics, Vol. 15, no. 1, , pp. 118-138, 2006.

[27] Groenen, P.J.F. and Velden, M. van de, Multidimensional Scaling. Econometric Institute Report EI, pp. 14-15, 2004.

[28] Pinker, S.,The Language Instinct: How the Mind Creates Language, Harper Collins, New York, 1994, http:www.isrl.uiuc.edu/ amag/langev/paper/pinker94theLanguage.html.

[29] R Development Core Team, R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, http://www.r-project.org, 2004.

[30] Therneau, T.M., Atkinson B., and R port by Brian Ripley, rpart: Recursive Partitioning. R package version 3.1-42, (http://www.mayo.edu/biostatistics), 2004.

[31] Ripley, B., class: Function for Classification. R package version 7.3-42, http://www.stat.ox.ac.uk/pub/MASS4/, 2010.

[32] UCI KDD archive.http://kdd.ics.uci.edu/databases/covertype/covertype.html, 2005.

[33] Kubat, M., and Matwin, S. Addressing the Curse of Imbalanced Data Sets: One Sided Sampling. In the Proceedings of the Fourteenth International Conference on Machine Learning, Nashville, pp. 179-186, 1997.

[34] Tatu, A., Albuquerque, G., Eisemann, M., Theisel, H., Magnor, M. and Keim, D., Combining automated analysis and visualization techniques for effective exploration of high-dimensional data. IEEE Symposium on Visual Analytics Science and Technology , pp. 59-66, 2009.

[35] Mollineda, R. A., Sanchez, J. S. and Sotoca, J. M., Data characterization for effective prototype selection. In: First edition of the Iberian conference on pattern recognition and image analysis, Lecture Notes in Computer Science, 3523, 27-34, 2005.

[36] Chawla, N., C4.5 and Imbalanced Data sets: Investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In Workshop on Learning from Imbalanced Datasets (ICML03), 2003.

[37] Wu, J., Xiong, H., Chen, J., COG: local decomposition for rare class analysis, Data Mining and Knowledge Discovery, Springer Netherlands Vol. 20, no. 2, pp. 191-220, 2010.

[38] Lim, M. K. and Sohn, S. Y. Cluster-based dynamic scoring model. Expert System with application, Vol. 32, pp. 427-431, 2007.

3.7 Appendix

3.7.1 R Codes

The `nearest ()` function is simply a wrapper for knn function. The purpose is to give the complexity "nearest" for each data point so we can use it as a measure of complexity of each class. The number of nearest neighbors is specified with the k= argument.

```
nearest <- function (X, n, k)
# Find k nearest neighbors of X[n, ] in the data frame or matrix
    X, utilizing function knn k-times.
{
    N <- nrow(X)
    # inds contains the indices of nearest neighbors
    inds <- c(n); i <- 0
    while (i < k) {
        # use knn1 for one index...
        j <- as.integer(knn(X[-inds, ],X[n, ], 1:(N-length(inds)
            )))
        # For faster computation use d as distance matrix then
            we can use min function for one index...
    j =as.integer(which.min(d[-inds,n]))
        # ...and change to true index of neighbor
        inds <- c(inds, setdiff(1:N, inds)[j])
        i <- i+1
    }
    # return nearest neighbor indices (without n, of course)
    return(inds[-1])
}
```

The following codes are for measuring Bayes error or Δ_μ the vector of mean differences for the classes.

```
  Dim.and.mean.diff.2.bayes.error<-function(Dim, mean.diff) {
# given dimension with means differing by mean diff each, what
    is bayes error rate?
  1-pnorm(sqrt(Dim*mean.diff*mean.diff)/2,0,1);
}

bayes.error.and.Dim.2.mean.diff<-function(bayes.error, Dim) {
# given dimension with bayes error, what is means differing by
    mean diff each?
  qnorm(1-bayes.error) * 2 * sqrt(1/Dim);
}
```

3.7.2 Results for 3 Dimensions:

The following tables give detailed analysis of simulation results for the three continuous variables case. These results are averages of 100 simulation results. Here n_1 and n_2 are the numbers of observations for class1 and class2, with Bayes Error representing the theoretical minimum error bounds, B1 and B2 are the estimated error bound by Monte carlo integration and where the Complexity(class)(k) for various classes, e.g, Complexity11 represent complexity for class 1 and k=1 for kNN approach, similarly Complexity21 represents, complexity for class 2 and k=1.

n1	n2	Bayes Error	B1	B2	Complexity11	Complexity21	Complexity13	Complexity23
1000	1000	0.4	401.77	399.38	472.98	472.07	459.82	462.32
1000	1000	0.3	297.91	299.83	388.54	387.88	358.89	357.47
1000	1000	0.1	98.13	101.39	148.5	147.84	122.23	121.22
1000	500	0.4	400.47	199.45	314.88	316.34	243.67	343.58
1000	500	0.3	300.69	148.97	265.59	267.27	207.75	273.44
1000	500	0.1	101.75	50.15	102.67	104.93	78.27	92.16
1000	350	0.4	397.66	139.8	245.2	247.27	163.72	275.04
1000	350	0.3	298.65	104.26	208.26	211.28	144.75	224.81
1000	350	0.1	99.1	35.43	82.04	84.03	59.64	76.94
1000	200	0.4	399.07	80.94	159.28	161.66	75.38	178.62
1000	200	0.3	298.35	59.37	140.35	142.95	77.28	154.82
1000	200	0.1	99.65	19.18	57.5	59.69	38.42	56.53
1000	90	0.4	0	90	81.15	80.96	21.2	87.33
1000	90	0.3	2.48	86.7	72.93	75.03	27.22	81.73
1000	90	0.1	13.13	32.87	34.04	37.02	19.27	36.5

Complexity15	Complexity25	Complexity17	Complexity27	Complexity19	Complexity29
451.43	454.24	444.67	447.37	441.3	442.49
342.46	343.67	334.57	334.68	328.34	328.1
114.6	113.93	110.79	110.13	108.76	108.55
205.81	360.01	178.38	372.12	157.92	381.62
181.47	276.89	166.71	279.13	155.81	280.92
70.45	89.2	67.24	86.54	65.02	86.09
119.49	290.97	92.82	301.1	74.56	308.73
117.8	230.8	101.71	234.94	91.86	237.44
53.02	74.52	50.04	73.85	47.83	73.77
42.13	187.28	24.34	191.66	15.32	194.2
53.7	160.57	41.76	164.34	34.64	167.47
33.47	55.76	30.59	55.36	28.68	55.82
6.65	88.89	2.48	89.58	0.99	89.87
13.14	84.74	7.82	86.31	4.83	87.18
15.26	37.26	13.18	38.02	11.84	38.59

This table shows the Mahalanobis Bounds, Bhattacharyya Lower and Upper Bound, for comparison with theoretical error bound and our complexity measurement in the above table.

Table 3.5: MB= Mahalanobis Bounds, BLB= Bhattacharyya Lower Bound and BUB= Bhattacharyya Upper Bound

n1	n2	Bayes Error	Mahalanobis	Bhattcharyya	N1	MB	BLB	BUB
1000	1000	0.4	0.128	0.032	0.980	0.484	0.375	0.484
1000	1000	0.3	0.550	0.140	0.915	0.440	0.253	0.435
1000	1000	0.1	3.285	0.824	0.678	0.275	0.051	0.219
1000	500	0.4	0.128	0.033	0.975	0.432	0.295	0.456
1000	500	0.3	0.550	0.139	0.915	0.396	0.214	0.410
1000	500	0.1	3.285	0.818	0.678	0.257	0.045	0.208
1000	350	0.4	0.128	0.034	0.978	0.375	0.234	0.423
1000	350	0.3	0.550	0.140	0.915	0.347	0.176	0.381
1000	350	0.1	3.285	0.819	0.678	0.236	0.039	0.193
1000	200	0.4	0.128	0.034	0.977	0.273	0.153	0.360
1000	200	0.3	0.550	0.142	0.912	0.258	0.119	0.323
1000	200	0.1	3.285	0.835	0.670	0.191	0.027	0.162
1000	90	0.4	0.128	0.038	0.971	0.150	0.076	0.265
1000	90	0.3	0.550	0.139	0.907	0.145	0.061	0.240
1000	90	0.1	3.285	0.823	0.671	0.121	0.015	0.121

Chapter 4

An Empirical Comparison of Under-Sampling Techniques in Relation to Data Complexity

This chapter studies the influence of both the imbalance ratio and the classifier on the performance of different under-sampling strategies to handle imbalanced data sets. A common approach to deal with the problem of imbalanced data sets is the use of a preprocessing step. In this study we analyze the usefulness of a data complexity measure associated with the minority class in order to predict the behavior of under-sampling methods. The study focuses on data complexity measure evaluation of how learning is affected when different re-sampling algorithms transform the original imbalanced data into artificially balanced class distributions. Experiments over 18 real data sets using seven different classifiers, ten re-sampling algorithms, six iterative sampling and three performance evaluation measures are studied to complete characterizations of the data sets and the differences among the different under-sampling results.

4.1 Introduction

In this Chapter, we will compare various under-sampling techniques to show that data complexity is useful to analyze the effect of the preprocessing in imbalanced data sets. We will identify which under-sampling method and classifier works better under different regions (groups) of data complexity space. Among the most intensely studied solution to the imbalance problem are data level solutions, which consist of artificially re-sampling the original data set until classes are approximately equally represented. The best class distribution for a data level solution for the class imbalance problem is still an open question and conclusions about it in the literature are divergent. Hulse et al. [1] suggest that the usefulness of each particular re-sampling technique depends on various factors, including the ratio between positive and negative examples, the characteristics of data, and the nature of the classifier. Several papers [2], [3], [4], [5] and [6] have also studied this dependence during the last decade. Nevertheless, their conclusions should be carefully interpreted because most of them are based on narrow learning frameworks.

This study significantly extends previous works by increasing the scope and detail at which is studied the influence of the imbalance ratio and the classifier on the effectiveness of the most popular under-sampling strategies. We will compare various under-sampling strategies and show that data complexity is useful to analyze the effect of preprocessing in imbalanced data sets. To this end, we will carry out a collection of experiments over 18 real databases with different levels of imbalance, employing seven classifiers, 10 re-sampling techniques and three performance metrics.

The rest of the chapter is organized as follows: Section 4.2 provides a brief overview of the existing under-sampling approaches and algorithms, in Section 4.3 various complexity measures are defined, in Section 4.4 the experimental set-up is described and results of our experiments are summarized, and Section 4.5 discusses the validity of the results. Finally, Section 4.6 gives conclusions and suggestions for future work.

Table 4.1: Under-Sampling Techniques

Methods	Algorithms	References	Pros	Cons
Random Under-sampling	Random Undersampling	Kubat et al, 1997 Japkowicz, 2001	Effective in correct class imbalance; Probability localized	Lost information
Neighborhood Cleaning Techniques	Tomek Link CNN Rule OSS NCL Rule	Tomek, I, 1976 Harts, P. E., 1968 Kubat et al., 1997 Wilson, D. L., 1972	Builds better classification Models	Computational expensive
Active Learning	Progressive Learning Adaptive Sampling Importance Sampling Selective Learning	John and lanley, 2003 Lyengar et al, 2000 Breiman et al, 1999 Ertekin et al, 2007	More efficient; No information lost No increase in computational cost	
Repetitive Under-Sampling	Easy Ensemble RUSBoost Cluster based Under-Sampling Classification using lOcal clusterinG(COG)	Lui et al. 2006 Seiffert et al., 2008 Lee et al, 2006 J. Wu et al., 2010 Zhang et al., 2010	More efficient	Increase in computational cost Depends on k

4.2 Current Under-Sampling (US) Approaches

Under-sampling has been the most investigated area in the classification, statistics, and data mining communities for dealing with imbalanced data. The main aim of under-sampling is to select a subset of majority class examples, eliminating other examples with least loss of information. Under-sampling can significantly improve the efficiency of the resulting classifier models and often builds a model that generalizes better to unseen points [6].

In the classification situation, under-sampling can be divided into four broad categories: Random under-sampling, Neighborhood cleansing techniques, Active Learning and Repetitive sampling methods. Table 4.1 provides a summary of under-sampling techniques used in classification, giving advantages and disadvantages of each technique, as well as references for the most common algorithms.

Weiss and Provost [7] noted that in many real data sets the original distribution of classes is not always the best distribution to use for a given classifier, and different re-sampling approaches try to modify the "original" distribution to another that is closer to the optimal one (i.e., class distribution size that favors all the classes in the data sets in terms of classification results). This can be done by over-sampling the minority class, under-sampling the

majority class, or combining simple over-and under-sampling techniques [8, 9]. All these strategies can be applied to any classifier, since these can be done at the data level, allowing the classifier to receive the training examples as if they belonged to a balanced data set. Thus any bias towards the majority class due to the different proportion of examples per class would be expected to be eliminated.

While these methods can result in greatly improved results over the use of original data set, they have also shown several important drawbacks. Under-sampling techniques may results in loss of valuable information. Moreover under-sampling of majority class modifies the prior probability of the classes, and can lead to a decrease in the accuracy of the negative class.

In order to minimize the loss of information, as randomly under-sampling may throw away valuable information, various Active Learning techniques have been investigated, from Neighborhood Cleansing techniques to Cluster-based under-sampling. We believe that the motivation behind the neighbor cleansing techniques is to reduce class overlap between the two classes. This may be the reason that most of the algorithms mentioned in Table 4.1 use 1-NN (kNN classifier, where k=1) to classify an example, and if a minority class example has its nearest neighbor from the majority class, it is eliminated. This algorithm is repeated a number of times, until no minority class examples are being misclassified. These methods however can seriously compromise the accuracy of majority class examples, if there is high overlap between the classes.

Active learning is phase-wise learning, where at each iteration a number of examples from the data set is added to the model. In these techniques all those examples which have been misclassified by a certain classifier are added to the final model [10]. These techniques requires a number of iterations of adaptive re-sampling which increases its computational cost. Furthermore this learning is tailored towards specific algorithms, hence a selected data set may not be ideal for other classifiers to build a model.

Repetitive under-sampling techniques are designed to alleviate some of the problems associated with random under-sampling (loss of information) [11]. Repetitive under-sampling constructs an ensemble of models, each using

a different sample from the majority class, hence reducing the loss of information when only one subset of the majority class is used. Cluster-based under-sampling technique is used in order to select majority class examples from different regions of the data set. In the clustering approach, the majority class is partitioned into k cluster. Each of the majority class partitions is combined with the entire minority class to create k training datasets, from which an ensemble of models is constructed. Clustering based under-sampling is more complex than random partitioning, as it depends on k, and how to fix k (number of clusters) is still a open question.

In this study we evaluate 11 different methods of under-sampling to balance the class distribution on the training data. In this section, we introduce those specific algorithms that we use later in our performance comparison. There are many under-sampling algorithms available in the literature. Here, several of the most popular algorithms are selected, with representation of all the categories mentioned in Table 4.1.

4.2.1 Random under-sampling (RUS):

This method aims to balance the data set by eliminating randomly selected examples of the majority class. Random under-sampling can throw away potentially useful information. The following under-sampling methods are devised to overcome the drawbacks of random under-sampling.

4.2.2 Neighborhood Cleansing Techniques (NCT)

4.2.2.1 Tomek links:

Tomek links [12] can be defined as follows: given two examples X_i and X_j belonging to different classes, denote by $d(X_i, X_j)$ the distance between X_i and X_j . An (X_i, X_j) pair is called a Tomek link if there is not an example X_l, such that $d(X_i, X_l) < d(X_i, X_j)$ or $d(X_j, X_l) < d(X_i, X_j)$. If two examples form a Tomek link, then either one of these examples is noise or both examples are borderline. Tomek links can be used as an under-sampling method or as a data cleaning method. As an under-sampling method, only examples be-

longing to the majority class are eliminated, and as a data cleaning method, examples of both classes are removed. This must be used with caution in a highly imbalanced data set in the presence of highly overlapped classes, since we may end up heavily reducing the majority class, hence accuracy of majority class will be seriously affected.

4.2.2.2 Condensed Nearest Neighbor Rule:

Hart's Condensed Nearest Neighbor Rule (CNN)[13] is used to find a consistent subset of examples. A subset $\widehat{X} \subseteq X$ is consistent with X if using a 1-nearest neighbor, $\widehat{X}$ correctly classifies the examples in X. An algorithm to create a subset $\widehat{X}$ from X as an under-sampling method is the following: First, randomly draw one majority class example and all examples from the minority class and put these examples in $\widehat{X}$. Afterwards, use a 1-NN over the examples in $\widehat{X}$ to classify the examples in X. Every misclassified example from X is moved to $\widehat{X}$. It is important to note that this procedure does not find the smallest consistent subset from X. The idea behind this implementation of a consistent subset is to eliminate the examples from the majority class that are distant from the decision border, since these sorts of examples might be considered less relevant for learning.

4.2.2.3 One-sided selection (OSS):

OSS [14] is an under-sampling method resulting from the application of Tomek links followed by the application of CNN. Tomek links are used as an under-sampling method to remove noisy and borderline majority class examples. Borderline examples can be considered as unsafe, since a small amount of noise can make them fall on the wrong side of the decision border. CNN aims to remove examples from the majority class that are distant from the decision border. The remaining examples, i.e. majority class examples, and all minority class examples are used for learning.

4.2.2.4 Neighborhood Cleaning Rule (NCL):

Neighborhood Cleaning Rule (NCL)[15] uses the Wilson's Edited Nearest Neighbor Rule (ENN)[16] to remove majority class examples. ENN removes any example whose class label differs from the class of at least two of its three nearest neighbors. NCL modifies the ENN in order to increase the data cleaning. For a two-class problem the algorithm can be described in the following way: for each example X_i in the training set, its three nearest neighbors are found. If X_i belongs to the majority class and the classification given by its three nearest neighbors contradicts the original class of X_i, then X_i is removed. If X_i belongs to the minority class and its three nearest neighbors misclassify X_i, then the nearest neighbors that belong to the majority class are removed. This may result in a poor model for the majority class, as all the majority class near to the class boundary will be eliminated.

4.2.3 Active Learning

4.2.3.1 Progressive Learning:

Provost et. al. [9] analyze a method for under-sampling by progressive sampling as long as model accuracy improves. They use a learning curve to decide when the model has converged (i.e., accuracy of the model does not change with increase of the training sets). A learning curve depicts the relationship between sample size and model accuracy. They analyze the efficiency of progressive sampling relative to the accuracy with all observations and conclude that a geometric sampling schedule (i.e., by increasing sample size in geometric progression) is asymptotically optimal. They provide empirical comparison between different sampling techniques and conclude that progressive sampling is most efficient. Although they did not use imbalanced data sets, we modify their algorithm to suit our problem of class imbalance. We start with all the examples from the minority class and a small subset of the majority class random selected, and keep increasing the sample size of majority class, until accuracy of the majority class is greater than the accuracy of the minority class. Hence we can determine the ideal class distribution of

the data sets. One of the disadvantages of this technique is its dependence on the classifier, i.e., the optimal class distribution for one classifier may not be ideal for another classifier.

4.2.4 Repetitive Under-Sampling

4.2.4.1 EasyEnsemble:

EasyEnsemble [17] repeats the random under-sampling approach n times, increasing the likelihood that each example will be included in the training of part of the ensemble. The author [17] used the balanced 50:50 class distribution, but any under-sampling approach can be used. After the under-sampling any classifier can be used, but the author used AdaBoost [18] as a classifier which itself creates an ensemble of models. As such, we also used AdaBoost in conjunction with a classification tree (CT).

4.2.4.2 RUSBoost:

RUSBoost [19] modifies the existing Ada.Boost.M2 [18] algorithm by performing under-sampling prior to construction of each model. A model is then constructed and, based on the model's performance, weights of the entire training datasets are modified as in the original AdaBoost Algorithm. The principle which we can see in this algorithm, is to include all those examples of the majority class which have been misclassified by the model. Hence a more efficient model can be built up. As far as weights of the minority class are concerned, modifying the weights does not have any real effect as the whole minority class will be used to form the training model.

4.2.4.3 Classification using lOcal clusterinG (COG):

Wu et al. [20], suggest a method to solve the minority class problem by Classification using lOcal clusterinG (COG). They cluster the majority class into small clusters using a standard k-mean clustering algorithm. An ensemble model is then constructed by classifying each majority class cluster against the minority class. The motivation behind their clustering idea is to get more

balance and to divide the complex concepts into simpler ones (i.e., non linear separable classes into linear separable class). This may not be appealing when there is a high degree of overlap between the class distributions, since their methodology will result in large misclassification error in either of the classes, and we may get better results for the minority at the expense of majority class.

4.2.5 Cluster-based Under-sampling

4.2.5.1 Clustering Undersampling:

Zhang et al., [21] propose a method of under-sampling method using clustering. They cluster majority class examples into k cluster using the k-means algorithm [22]. Suppose the number of majority class is N_{maj} and for each cluster we have the number of examples as N_{maj_i} for $1 \leq i \leq k$, then let the proportion of majority class in each cluster be $r_i = N_{maj_i}/N_{maj}$. From each cluster the number of examples for majority class will be selected as $s_i = N_{min} * r_i$ where N_{min} is the number of minority class examples. The motive behind this technique is to get a balanced distribution to minimize the loss of information due to under-sampling, and to select the majority class examples from different regions of data sets.

The effectiveness of these re-sampling approaches has been investigated in previous studies with respect to different data sets and classification models. However, most of them have focused on some specific learning factors (classifiers, data sets, performance metrics, re-sampling strategies), but disregarding the effect of others.

- Japkowicz and Stephen [5] discussed the performance of basic resampling methods when using a C5.0 decision tree (Classification Tree) induction system over a reduced number of artificial and real-world data sets. The error rate on each class was recorded to carry out this study.

- Drummond and Holte [6] presented the performance of basic re-sampling

methods when using a C5.0 decision tree over four real data sets. Ten fold cross validation estimated the mean performance of the classifier.

- Barandela et al. [23] investigated several under- and over-sampling techniques. The experiments were constrained to five real data sets using the nearest neighbor rule for classification and the geometric mean as the performance evaluation metric.

- Estabrooks et al. [2] studied random strategies at different resampling rates with C4.5 classifiers. They evaluated the performance on seven artificial and five real data sets by means of the overall error rate and the error on each class.

- Batista et al. [24] conducted a broad experimental analysis with 13 databases and 10 resampling methods, but conclusions were limited to the C4.5 decision tree and the use of the area under the ROC curve for assessing the results.

- Hulse et al. [1] addressed the imbalance problem with 35 different data sets, seven sampling techniques and 11 commonly-used learning algorithms using four evaluations matrices for assessing the results.

4.3 Empirical Study:

To relate the performance of existing under-sampling methods to data complexity with real data, we have used 18 data sets from the UCI data repository [25]. For every binary data set generated we computed the data complexity measure over the complete data set before the processing and splitting of the data. CM as define in Chapter 3 is our Complexity Measurement for the minority class, whereas F1, L3 and N1 are the complexity measurements proposed by Ho [26]; IR (Imbalance Ratio) defined as the number of negative class examples, divided by the positive class examples is usually used to show the degree of imbalance [1]. We use CM as the general indicator of whether a data set is easy or difficult to learn, and the examples are arranged

accordingly, due to fact that it has been shown to be highly correlated with the performance of the classifier on a minority class, in comparison to any other complexity measurement.

Table 4.2: Description of UCI data sets

Data Set	Size	Target	Size of Min	Size of Maj	%Min	IR	F1	L3	N1	CM
Vehicle3	846	bus	218	628	25.8	2.88	0.169	0.368	0.049	0.037
WDBC	569	malignant	212	357	37.3	1.68	3.568	0.007	0.831	0.080
Yeast4	1484	ME1	44	1440	3.0	32.73	4.198	0.500	0.122	0.273
Yeast2	1484	ME3	163	1321	11.0	8.10	2.751	0.500	0.679	0.325
Satimage	6435	class 4	626	5809	9.7	9.28	0.375	0.500	0.556	0.345
Haberman	306	class 2	81	225	26.5	2.78	0.185	0.497	1.024	0.346
Ionosphere	351	bad	126	225	35.9	1.79	0.609	0.980	1.129	0.405
Pima	768	class 1	268	500	34.9	1.87	0.576	0.500	1.008	0.427
Vehicle2	846	saab	217	629	25.7	2.90	0.381	0.231	0.284	0.452
Yeast3	1484	exc	35	1449	2.4	41.40	2.302	0.500	0.594	0.486
Abalone2	4177	Ring<7	448	3729	10.7	8.32	2.947	0.520	0.365	0.489
Yeast1	1484	nuc	429	1055	28.9	2.46	0.242	0.500	0.879	0.564
Vehicle1	846	opel	212	634	25.1	2.99	0.186	0.351	0.936	0.566
Cmc	1473	class 2	333	1140	22.6	3.42	0.245	0.500	0.908	0.718
WPBC	198	recur	47	151	23.7	3.21	0.142	0.780	0.931	0.745
Abalone1	4177	Ring=7	391	3786	9.4	9.68	0.879	0.650	0.489	0.903
Balance	625	Balance	49	576	7.8	11.76	0.001	0.500	0.971	1.000
Abalone3	4177	Ring=19	32	4145	0.8	129.53	0.530	0.500	0.731	1.000

4.3.1 Learning algorithms

We have tried to tackle two-class classification problems from a wide perspective taking into account different supervised techniques existing in the machine learning field, leading us to consider four base classification models, belonging to very different approaches: Logistic Regression (LR), Classification Trees (CT), Artificial Neural Networks (ANN) and Support Vector Machines (SVM). For more detail how each particular classification approach works, see Appendix 2.7.

LR is one of the most used statistical models to predict binary outcomes and has interesting properties (see Appendix 2.7) concerning the interpretation of the coefficients [27]. CT has connections with the Social Sciences [28] and the Artificial Intelligence (AI) fields [29]; CT stand out for the ease of

interpretation of the obtained model. The Artificial Intelligence (AI) community has developed a powerful computational paradigm, the ANN [30], which has been progressively incorporated into statistical practice [31, 32], but their black-box nature makes the interpretation of the resulting models very difficult. SVM emerged from Statistical Learning Theory, or Vapnik-Chernovenkis theory [33], which is one of the most investigated topics in machine learning. An excellent survey of SVMs can be found in [34]. The four models are freely available in the R system [35] which also provides the user with a powerful statistical programming language. Ihaka and Gentleman [36] present an introduction to the main characteristics of the R system. The programming resources of this system are very suitable for programming ensemble methods, where a certain number of models are constructed by resampling the set of cases and/or the set of predictors, aggregating the models by majority voting (see Appendix 4.6).

4.3.1.1 Ensemble classification methods

Several methods to combine different classification models have been proposed. They are based on the combination of models fitted from samples and sets of variables generated from the original data set. We consider the following ensemble methods: Random Forests, Bagging and Boosting. Bagging and boosting are meta-algorithms that pool decisions from multiple classifiers.

Random Forests have been proposed by Breiman [37] as a way to combine many different trees. A number of trees are constructed. Each one is grown over a bootstrap sample of the training data set, and a random selection of variables is considered to choose splits in each node. As in bagging, the trees are combined by majority voting. Bagging [18] is a voting method that uses slightly different training sets (generated by bootstrap) to make different base learners, whereas the Boosting algorithm [18] aims to improve the classification accuracies of any "weak" learning algorithm. It weights each sample reflecting its importance and places more weight on those examples which are more often misclassified. This forces the learners to emphasize

those samples that are hard to correctly classified.

4.3.1.2 Clustering

Clustering has been recognized as a useful spatial data mining method. In statistics as well as in machine learning many efforts exist to identify and to describe groups of similar objects [38]. The key idea to utilize clustering techniques for under-sampling purposes, is to select only a few representatives from each cluster of similar objects (majority class). This set of representatives then forms the output of majority class selection. In this study we use the CLARA algorithm, where CLARA is a sampling-based algorithm which implements Partition around Mediods (PAM) on a number of sub-data sets [39]. i.e., CLARA draws a sample of the data set, applies PAM on the sample, and finds the medoids (the most centrally located object) of the sample. The point is that, if the sample is drawn in a sufficiently random way, the medoids of the sample would approximate the medoids of the entire data set. To come up with better approximations, CLARA draws multiple samples and gives the best clustering as the output. Here, for accuracy, the quality of a clustering is measured based on the average dissimilarity of all objects in the entire data set, and not only of those objects in the samples. Experiments [39] indicate that five samples of size $40 \pm 2k$ give satisfactory results. This allows for faster running times when the number of observations is relatively large.

4.3.2 Study Design

All the preprocessing (under-sampling) techniques considered for our experiment were employed on each of the training partitions in order to reduce the effect of class imbalance. In total, five-fold CV runs times 18 data sets gives 90 different training/testing data sets, then 13 under-sampling techniques plus raw data sets were used for each of the 90 training datasets, resulting in $90 \times 13 = 1170$ data sets, each of which is used for classification. Since we used 7 learning algorithms, a total of $1170 \times 7 = 8190$ models were constructed and considered in this study for Random Under Sampling and Neighborhood

Cleansing Techniques (NCT).

We consider two random under-sampling ratios, i.e., balanced class distribution and minority class as 35% of the whole training set. The purpose of considering only two under-sampling ratios is to keep the study simple. Other studies [1, 3] have considered many imbalance ratios. However, we found that these two ratios were adequate to learn about the minority class in highly imbalanced data sets. Moreover we have investigated optimal class distribution other than these two (balance and 35%) by Learning Curve (LC) as described in section 4.2.3.

For ensemble models we use ten iterations, with ten repetitions of under-sampling technique, resulting in 100 models being constructed for one data set, as proposed in [19].

We use Sensitivity (the accuracy of the minority class) to measure the performance of a classifier. Sensitivity is chosen because we are most interested in accuracy for the minority class rather than overall accuracy. We also evaluate the algorithms using the metric G-Mean defined as $\sqrt{TP \times TN}$ [40], which corresponds to the geometric mean between the correct classification rates for positive (sensitivity) and negative (specificity) examples, respectively. We also use the Area Under the ROC (receiver operating characteristic) curve (i.e., AUC) [41] to evaluate the selected classification models. AUC is considered the preferred criterion when interest is the minority class [5].

All analyses were done using 5-fold cross validation, i.e., a random partition of the data into five sets of 20%, with the combination of 4 of them (80%) as training and the remaining one as the test set. For each data set we consider the average results of the five partitions. This partition was done as stratified (class based partition) to preserve the original imbalance ratio between the classes.

All programs were written in the statistical environment R [35] (for relevant R codes see Appendix 4.6). For the construction of tree models we used the `RPART` package [42]; for nearest neighbor based on Euclidean distance we used R package `class` [43]; with a little modification of the input and output to compute Random Forest distances we used the R package `randomForest` [37]; for svm and glm we use R package `e1071` [44], and for Bagging and

Boosting we use R package `adabag` [45]. For comparison among different classifier we conducted the following statistical test.

4.3.3 Statistical Analysis: Friedman Test

The Friedman test [46, 47] is a non-parametric equivalent of the repeated-measures Analysis of Variance (ANOVA). It ranks the classifier for each data set separately, the best performing classifiers getting the rank of 1, the second best rank 2, and so on. In case of ties average ranks are assigned. Let r_i^j be the rank of the j-th of k classifier on the i-th of N data sets. The Friedman test compares the average ranks of classifiers, $R_j = \frac{1}{N}\sum_i r_i^j$. Under the null-hypothesis, which states that all the classifier are equivalent and so their ranks R_j should be equal, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{4.1}$$

is distributed according to χ_F^2 with $k-1$ degrees of freedom, when N and k are big enough (as a rule of a thumb, $N > 10$ and $k > 5$). For a smaller number of algorithms and data sets, exact critical values have been computed [48, 49].

Iman and Davenport [50] showed that Friedmans χ_F^2 is undesirably conservative and derived a better statistic

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2} \tag{4.2}$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. The table of critical values can be found in any statistical book.

As for the two-classifier comparisons, the (non-parametric) Friedman test has theoretically less power than (parametric) ANOVA when the ANOVA's assumptions are met, but this does not need to be the case when they are not.

4.3.4 Results:

In our analysis only F1, L3 and N1 measure of the 12 measure presented in Table 3.1 proved to be more correlated with the sensitivity in our data sets. Figure 4.1 shows the relationship between the different complexity measures and the sensitivity values (average sensitivity value from different classifiers). There is high correlation between the sensitivity and CM values. For all the raw data sets, pooling results from the different classification techniques used, empirically, pooling (ensembles) tend to yield better results when there is a significant diversity among the models [51, 52]. Many ensemble methods, therefore, seek to promote diversity among the models they combine [53]. We found the correlation between the complexity measurement and the sensitivity values, for CM, IR, F1, L3, and N4 to be -0.90, -0.41, 0.42, -0.27 and -0.27 respectively.

Hulse et al. [1] suggested grouping datasets using percentage of minority class as thresholds to show the effectiveness of the base classifier and the effectiveness of preprocessing techniques for the different imbalance scenarios. We will follow their approach, but using CM thresholds to define groups. Since CM is highly correlated with the sensitivity values, this makes CM an obvious candidate for defining thresholds. In our case then we are using CM thresholds to characterize the effectiveness of the different models and under-sampling techniques. We choose four CM ranges for our comparisons, defined as: CM $\leq$ 20%, 30% $<$ CM $\leq$ 40%, 40% $<$ CM $\leq$ 50% and CM $>$ 50% respectively. An initial analysis was carried out on the data grouped by complexity (CM), into the categories presented in Table 6.2. One range of CM values was not found in the data sets we evaluated: 20% $<$ CM $\leq$ 30% did not occur.

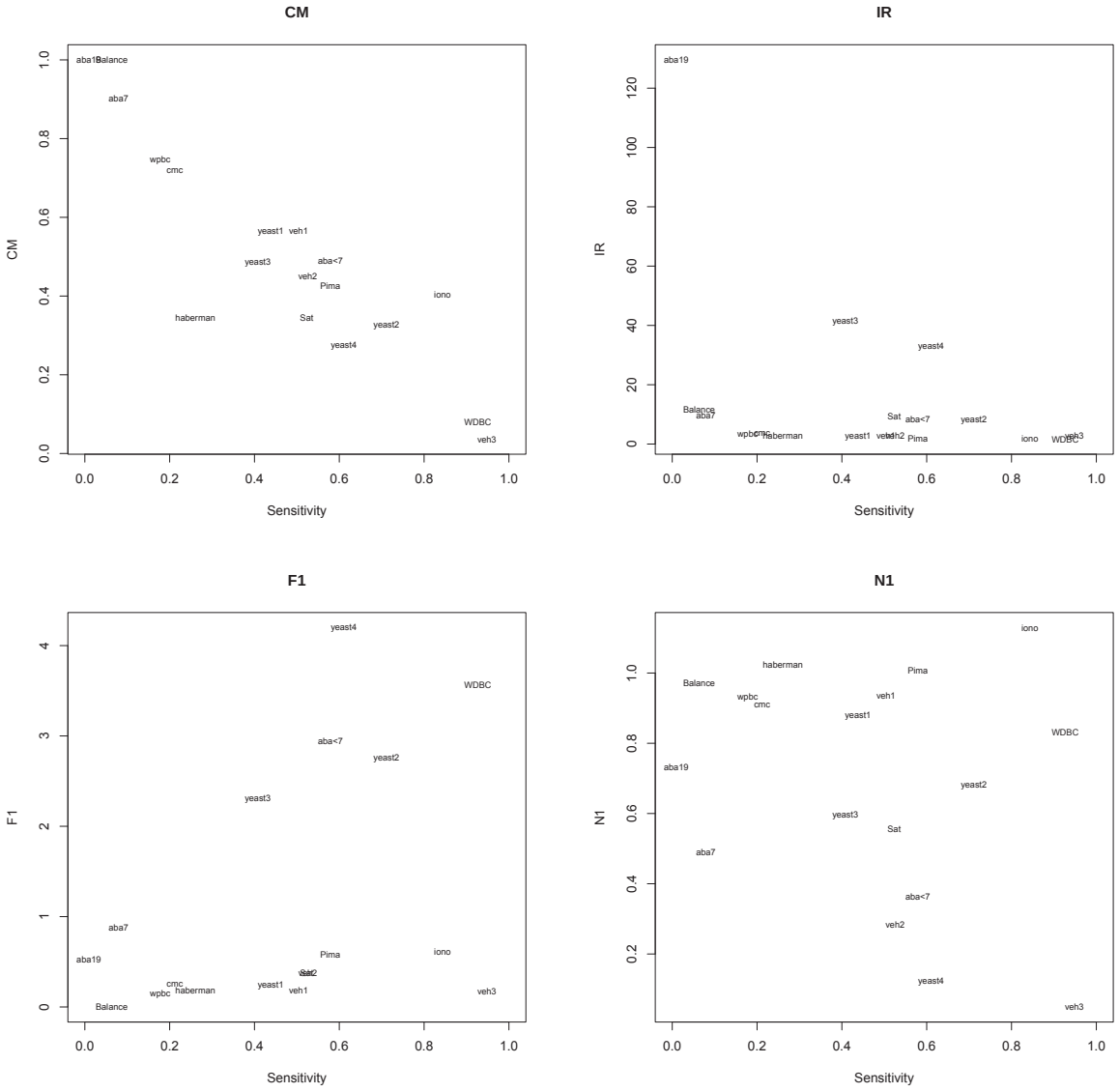

Figure 4.1: Scatter plot for Sensitivity values (average sensitivity value from different classifiers) and Complexity Measurement for UCI data set, top left is our proposed Complexity Measurement (CM), top right is Imbalance Ratio (IR), bottom left is Fisher Discriminant Ratio (F1) and bottom right shows Fractions of points on class boundary (N1). Every data set is represented by its name

Table 4.3: Results for Raw Data Sets

Classifier	30%-40%				41%-50%				>50%			
	GM(SD)	Sen(SD)	Spe(SD)	AUC(SD)	GM(SD)	Sen(SD)	Spe(SD)	AUC(SD)	GM(SD)	Sen(SD)	Spe(SD)	AUC(SD)
CT	0.720(0.12)	0.567(0.13)	0.926(0.11)	0.746(0.12)	0.735(0.11)	0.620(0.17)	0.880(0.06)	0.750(0.09)	0.411(0.09)	0.270(0.12)	0.878(0.08)	0.574(0.06)
RF	0.702(0.16)	0.528(0.2)	0.965(0.05)	0.747(0.12)	0.736(0.12)	0.600(0.17)	0.922(0.07)	0.761(0.1)	0.316(0.22)	0.187(0.18)	0.958(0.03)	0.572(0.08)
SVM	0.656(0.26)	0.509(0.26)	0.973(0.02)	0.741(0.14)	0.714(0.14)	0.563(0.20)	0.941(0.06)	0.752(0.11)	0.243(0.24)	0.126(0.18)	0.976(0.03)	0.551(0.07)
NN	0.703(0.16)	0.539(0.2)	0.955(0.04)	0.747(0.12)	0.720(0.1)	0.575(0.17)	0.932(0.07)	0.753(0.08)	0.411(0.29)	0.291(0.24)	0.913(0.1)	0.602(0.11)
LR	0.517(0.29)	0.354(0.29)	0.855(0.24)	0.605(0.24)	0.748(0.08)	0.618(0.13)	0.933(0.05)	0.775(0.05)	0.205(0.25)	0.114(0.17)	0.818(0.42)	0.466(0.25)

Table 4.4: Best under-sampling technique by Geometric Mean

Classifier	30%-40%				41%-50%				>50%			
	RUS		NCT		RUS		NCT		RUS		NCT	
	technique	mean(sd)	technique	mean(sd)	technique	mean(sd)	technique	mean(sd)	technique	mean(sd)	technique	mean(sd)
CT	Balanced	0.814(0.188)	NCL	0.780(0.116)	LC	0.779(0.089)	NCL	0.777(0.064)	Balanced	0.641(0.088)	NCL	0.536(0.195)
RF	Balanced	0.835(0.160)	NCL	0.801(0.090)	Balanced	0.839(0.071)	NCL	0.796(0.082)	Balanced	0.666(0.119)	NCL	0.479(0.337)
SVM	Balanced	0.838(0.160)	NCL	0.800(0.105)	Balanced	0.841(0.083)	NCL	0.811(0.077)	Balanced	0.680(0.114)	NCL	0.496(0.289)
NN	35%	0.806(0.178)	NCL	0.801(0.109)	Balanced	0.827(0.061)	Tomek	0.761(0.082)	Balanced	0.721(0.108)	NCL	0.586(0.265)
LR	Balanced	0.759(0.163)	Tomek	0.691(0.109)	LC	0.667(0.375)	NCL	0.609(0.351)	Balanced	0.558(0.269)	NCL	0.344(0.327)

Table 4.5: Best under-sampling technique by Sensitivity

Classifier	30%-40%				41%-50%				>50%			
	RUS		NCT		RUS		NCT		RUS		NCT	
	technique	mean(sd)	technique	mean(sd)	technique	mean(sd)	technique	mean(sd)	technique	mean(sd)	technique	mean(sd)
CT	Balancced	0.808(0.201)	NCL	0.725(0.072)	Balanced	0.784(0.090)	NCL	0.724(0.103)	Balanced	0.658(0.054)	NCL	0.466(0.267)
RF	Balancced	0.838(0.176)	NCL	0.715(0.063)	Balanced	0.850 (0.055)	NCL	0.750(0.145)	Balanced	0.689(0.159)	NCL	0.455(0.341)
SVM	Balancced	0.840(0.196)	NCL	0.729(0.077)	Balanced	0.862 (0.068)	NCL	0.765(0.131)	Balanced	0.716(0.195)	NCL	0.436(0.317)
NN	Balancced	0.783(0.204)	NCL	0.737(0.087)	Balanced	0.840 (0.063)	NCL	0.706 (0.137)	Balanced	0.762(0.136)	NCL	0.558(0.273)
LR	Balancced	0.697(0.224)	Tomek	0.627(0.251)	LC	0.652 (0.357)	NCL	0.562 (0.349)	Balanced	0.588(0.279)	OSS	0.376(0.321)

Table 4.6: Best under-sampling technique by AUC

Classifier	30%-40%				41%-50%				>50%			
	RUS		NCT		RUS		NCT		RUS		NCT	
	technique	mean(sd)	technique	mean(sd)	technique	mean(sd)	technique	mean(sd)	technique	mean(sd)	technique	mean(sd)
CT	Balanced	0.816(0.184)	NCL	0.791(0.116)	LC	0.781(0.088)	NCL	0.786(0.061)	Balanced	0.646(0.085)	NCL	0.617(0.089)
RF	35%	0.841(0.168)	NCL	0.810(0.092)	Balanced	0.840(0.070)	NCL	0.809(0.068)	Balanced	0.670(0.118)	NCL	0.629(0.117)
SVM	Balancced	0.843(0.153)	NCL	0.810(0.103)	Balanced	0.842(0.083)	NCL	0.821(0.073)	Balanced	0.692(0.104)	NCL	0.624(0.110)
NN	35%	0.822(0.153)	NCL	0.810(0.107)	Balanced	0.829(0.060)	NCL	0.791(0.046)	Balanced	0.725(0.106)	NCL	0.669(0.094)
LR	Balancced	0.769(0.153)	Tomek	0.734(0.085)	35%	0.671(0.375)	NCL	0.627(0.351)	Balanced	0.564(0.267)	Tomek	0.483(0.219)

Table 4.7: Results for Repetitive Techniques

Classifier	30%-40%				41%-50%				>50%			
	GM(SD)	Sen(SD)	Spe(SD)	AUC(SD)	GM(SD)	Sen(SD)	Spe(SD)	AUC(SD)	GM(SD)	Sen(SD)	Spe(SD)	AUC(SD)
Bagging	0.689(0.17)	0.525(0.19)	0.948(0.07)	0.737(0.13)	0.732(0.09)	0.595(0.14)	0.921(0.07)	0.758(0.08)	0.346(0.28)	0.210(0.21)	0.939(0.04)	0.575(0.09)
Boosting	0.746(0.17)	0.610(0.2)	0.929(0.09)	0.770(0.15)	0.744(0.1)	0.605(0.15)	0.932(0.06)	0.769(0.09)	0.365(0.27)	0.232(0.2)	0.919(0.05)	0.575(0.08)
Easy Ensemble	0.854(0.14)	0.886(0.13)	0.827(0.15)	0.856(0.14)	0.833(0.08)	0.839(0.07)	0.829(0.09)	0.834(0.07)	0.701(0.10)	0.748(0.12)	0.664(0.08)	0.706(0.10)
RUSBoost	0.849(0.15)	0.876(0.14)	0.831(0.14)	0.854(0.14)	0.831(0.07)	0.833(0.06)	0.832(0.09)	0.832(0.07)	0.674(0.13)	0.764(0.09)	0.613(0.16)	0.689(0.11)
CLUS(bal)	0.841(0.14)	0.863(0.13)	0.824(0.15)	0.844(0.14)	0.826(0.08)	0.815(0.06)	0.839(0.10)	0.827(0.08)	0.615(0.13)	0.606(0.19)	0.689(0.13)	0.647(0.08)
CLUS(ind)	0.504(0.30)	0.994(0.01)	0.330(0.36)	0.662(0.18)	0.474(0.20)	0.970(0.06)	0.277(0.19)	0.624(0.08)	0.308(0.19)	0.944(0.07)	0.165(0.10)	0.555(0.05)

Table 4.3 presents a summary of the results obtained by the learning algorithms on the different categories of problems. In the columns we show the different CM groups we used in this study, and for every group various evaluation matrices have been calculated. We can visualize performance of the different classifiers presented in Table 4.3 from Fig. 4.2. The diagrams in Fig.4.2 present the performance of the different classifiers, under different evaluation metrics along with their 95% confidence interval (shown by an error line at the top of each bar), on the problem categories which affect their learning capacity. The accuracy alone is not a good measure of performance. The analysis should instead focus on the following criteria: high values for Sensitivity, GM, and AUC indicate a good classification, while high Specificity values in comparison to Sensitivity values reveal a classification which is biased towards the majority class. Moreover, the larger the difference between the Specificity and the Sensitivity, the more biased the classification process is. This is the reason we are considering Sensitivity, GM and AUC values to make the following comments. Even though we are not interested in specificity and overall accuracy of the data sets, we can't disregard these factors as they show the biases of different under-sampling technique against majority class. The results agree with the different imbalances concept presented in [54]. The value of the CM plays an important role in the performance of the classifiers. An increase in complexity leads to classifier performance degradation on the minority class; while one would expect that high complexity significantly affects the capacity of classifiers to achieve acceptable performance scores, it must be kept in mind that CM is independent of any classifier and evaluation measure used. As it can be observed from Fig. 4.2 the behavior of classifiers on less complex data sets is better and more uniform than on categories of problems of higher complexity: in group CM <20 almost all classifiers seem to be robust to the imbalance problem, with the uniform 95% confidence interval across the data sets used for this group. With high complexity shown by the various CM groups all the classifiers are behaving in a similar way across various evaluation measures, with uniform 95% confidence interval. We carried out Friedman test to see any significant differences among classifiers between datasets within the

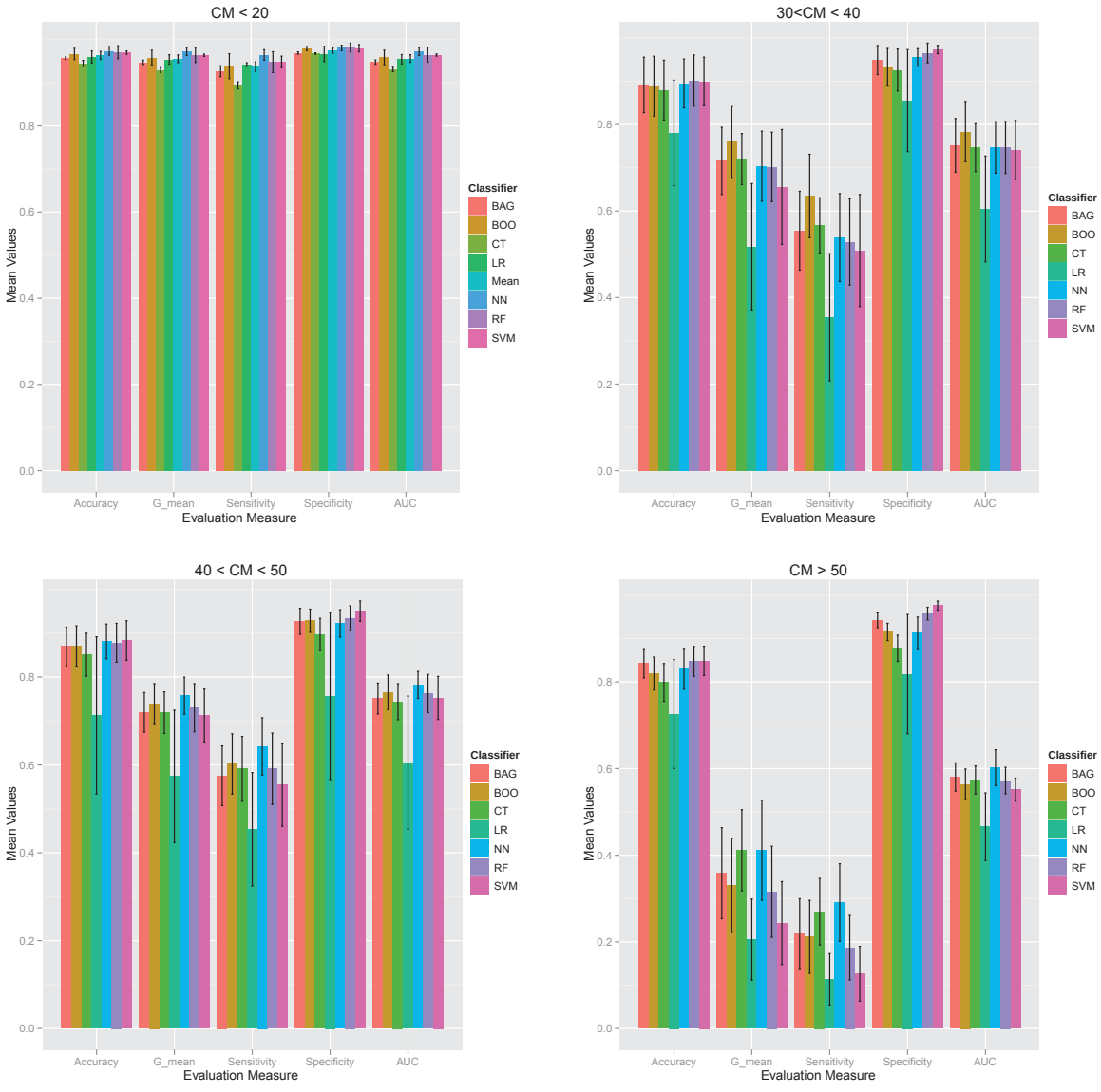

Figure 4.2: A summary of the results obtained by the learning algorithms on the different categories of problems. Each panel shows the results for a different range of complexity (CM) with each classifier evaluated by five different measures along with their 95% confidence interval (shown by an error line at the top of each bar).

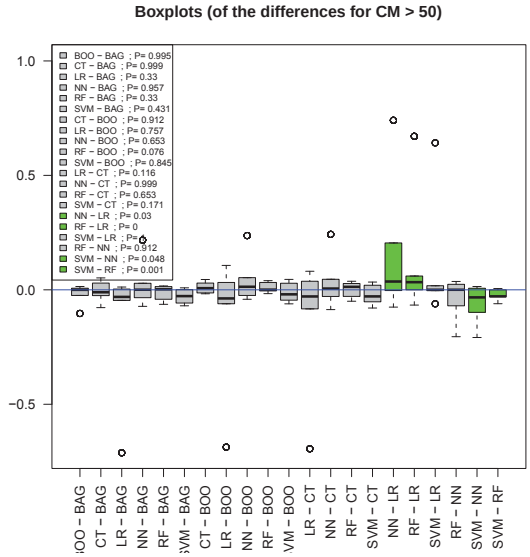

Figure 4.3: A difference obtained by the learning algorithms on the different data sets for group CM > 50. The result for a differences of AUC values among the classifier evaluated by Friedman Test. On x-axis all possible combination of difference between classifier, where as in legend relative p-values of post hoc test is given.

group on the value of AUC. We found significant difference among classifiers; by the posthoc test we found that LR is significantly different for the CM groups i.e, $30\% < \text{CM} \leq 40\%$ and $40\% < \text{CM} \leq 50\%$ and LR and SVM for the group CM > 50 (shown graphically in Fig 4.3). SVM performance rapidly degrades (increasingly difference between sensitivity and specificity) with increasing complexity, which shows that SVM is sensitive to complexity group with class imbalance. LR in most of the cases performed poorly. This is the reason that we find Friedman's test for various classifier, within the various complexity groups, statistically significant.

For illustration Friedman test for differences among classifiers on AUC value between different data sets for CM > 50 is shown in Fig 4.3. From Fig 4.3 it is clear that Friedman's test is significant due to poor performance of LR and SVM and in comparison to the NN and RF. The rest of the classifier are not significantly different.

The results prove that the learning capabilities of the classifiers considered are affected by an increased CM. It can be observed that Neural Network

(NN) is generally more robust than Classification Tree (CT) to the imbalance problem. Moreover, NN achieves relatively high Sensitivity, GM and AUC values in most cases (in higher CM groups), and NN yields at least the second best performance in all cases, which makes it the most robust out of all the classifiers evaluated. None of the LR results were relatively good in any of the cases studied, which makes it suitable only for baseline problem assessment.

The above observations provide an affirmative answer to one of the open questions in [54] (their study found that NN is the most robust classifier under different class imbalances domains), namely whether the conclusions presented there can be applied to real-world domains.

However, our results also indicate that SVM is the most sensitive to CM groups (shown by the increasingly difference between sensitivity and specificity by various CM groups). This means that, for the particular case of SVMs, the conclusion drawn from experiments on synthetic data cannot be extended to real data sets. A reason for this could be the following: in the case of synthetic data sets, even for large Imbalance Ratios (IRs), the examples which represent reliable support vectors are present in the data, due to the systematic data generation process, while in the case of real problems, these vital learning elements might be missing. This makes SVMs the weakest classifiers in most real-world imbalanced problems.

The results suggest that our complexity measure (CM) represent a good (i.e. monotonic) indicator of influence on accuracy of the minority class in the classification process. A relation between complexity and classifier performance is revealed, i.e. the higher the complexity, the greater the performance degradation. This suggests that complexity is a useful concept to explain and predict classifier robustness when faced with imbalance problems in real world situations.

We observed that for CM < 20, all the classifiers perform well, so this group is not considered further. Tables 4.4, 4.5, 4.6 depict the average results for GM, Sensitivity and AUC for various undersampling techniques and Neighborhood cleansing Techniques (NCT) as described in section 4.2.1, from the four base classifier i.e., CT, LR, NN, and SVM and ensemble classifier RF for all the data in a certain CM group. Results for the best Sensitivity, Geo-

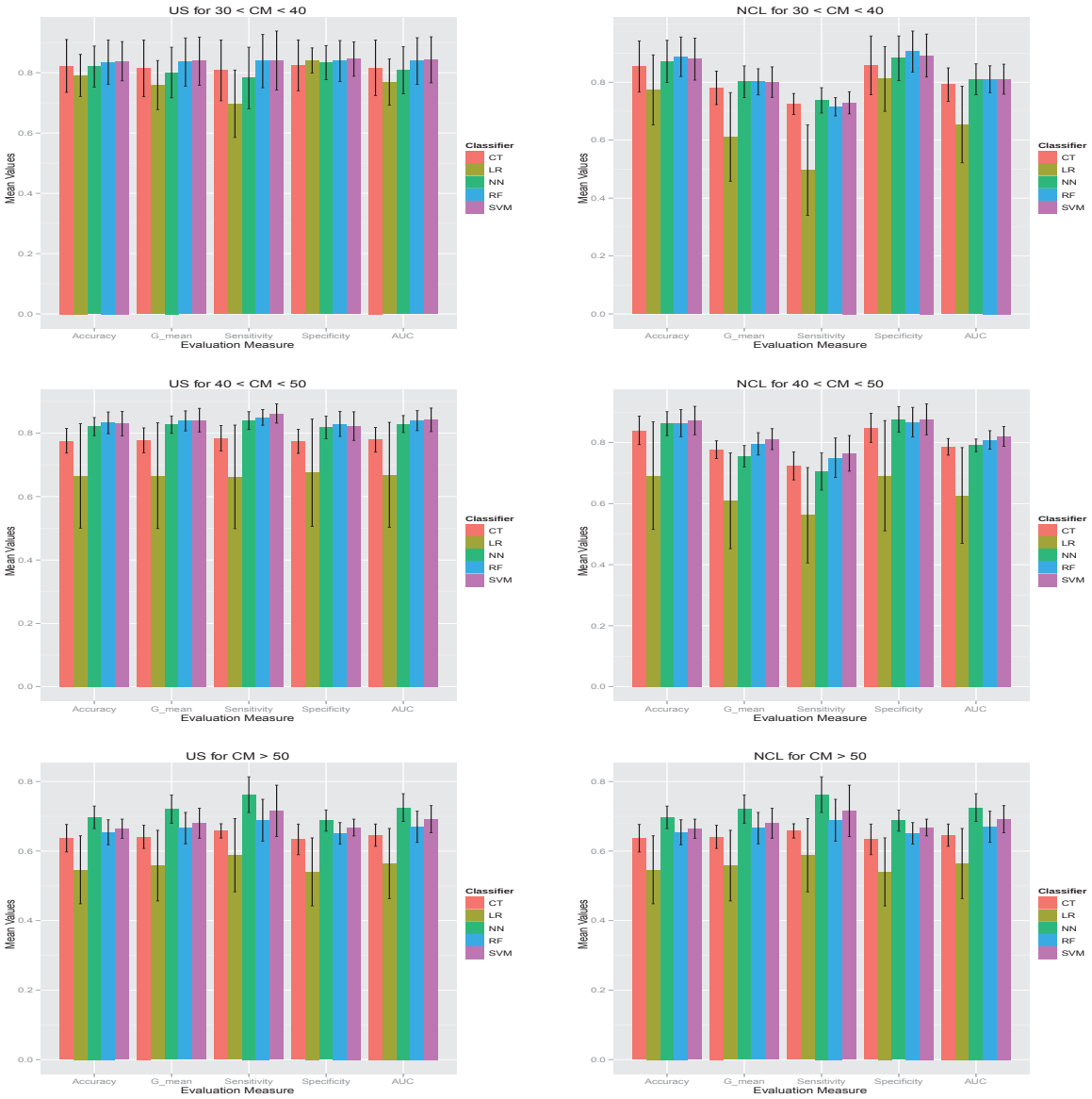

Figure 4.4: A summary of the results obtained by the learning algorithms on the different categories of problems using the under-sampling technique that gave the best sensitivity along with their 95% confidence interval (shown by error line at top of each bar). The left hand graphs represent the best random under-sampling techniques, whereas right hand shows best NCT techniques.

metric Mean and AUC have been used to select which tecnique/methodology performs best within each CM group. The following abbreviation has been used in the tables for represent the best technique: for RUS, Balanced means random under sampling (RUS) is done on the majority class to achieve balanced class distribution for training data, 35% represent that minority class is 35% of the whole training data sets, and LC (Learning Curve) as explained in section 4.2.3. Whereas for the NCT, various abbreviations is explained in section 4.2.2. It is interesting to note that in most of the cases if the classifier has high sensitivity value, it will be accompanied by high GM and AUC values. It is understandable that high GM and AUC can only be achieved if we have high sensitivity and specificity values. We can see that all the base classifiers prefer a balanced distribution, which implies that due to the increased degree of class learning difficulty (represented by our CM values) accompanied by the class imbalance, classifiers do need a reasonable amount of examples from the minority class to build a model on it. But it comes at the cost of accuracy of the majority class. For the neighborhood cleaning techniques (NCT), NCL proves to be the best techniques. This is due the fact that this technique removes all the majority class examples for the training data sets that are misclassified by the k nearest neighbor (k=3), as it forces the decision boundary to favor the minority class. This is an effective technique if there is low overlap between the classes. This technique should be handled with care when we have severe imbalance along with high overlap between the classes, as it may result in heavily reducing the majority class examples which will result in a poor classification model for the majority class.

The results for Tables 4.4, 4.5, 4.6 has been summarized as follows.

- From Table 4.3 for complexity 30%<CM≤40%, we have average specificity of 94% and sensitivity of 53% across all the data sets. Using RUS (Balanced) techniques, the specificity decreases by 11%, with the increase in sensitivity of 27%; for the NCL, specificity decreases around 9% while the increase in sensitivity is around 22%. Similarly increases for the GM for Balanced and NCL are 12% and 9% respectively, and for AUC 9% and 6% respectively.

- For complexity 40%<CM≤50% (Table 4.3) we have average specificity of 90% and sensitivity of 57% across all the data sets, which shows the increased degree of overlapped between the classes. The average decrease in the specificity for a balanced training set is around 11% with the increase in the sensitivity of 22%; for the NCL decrease in the specificity is around 9% with the increase in sensitivity of 13%. Similarly increase for the GM for balanced and NCL are 8% and 4%, and AUC 6% and 3% respectively.

- For complexity > 50% Table 4.3 we have average specificity of 91% and sensitivity of 20% across all the data sets. These data sets are extremely difficult to learn for the minority class and any under-sampling attempt in improving the accuracy of minority class will result in severe decrease in the accuracy of majority class. This can be judged from Table 4.5, 4.4, 4.6: specificity decreases by 29% with the increase in sensitivity of 48%; for the NCL decrease in specificity is around 39% with an increase in sensitivity of around 26%. Similarly increase for the GM for balanced and NCL are 32% and 16% respectively, and for AUC 10% and 5%. NCL is not giving compatible results in these cases in due the fact, that because of high overlap between the classes, this techniques is severely under-sampling the majority class. For higher CM value there is a compromise between the sensitivity and specificity, so in order to increase the classification accuracy of minority class, we find a high decrease in the classification accuracy of majority class.

For illustration, Figure 4.4 gives the results for the RUS (Balanced, left panels) or NCT (NCL, right panel) which produced the best sensitivity values along with 95% confidence interval shown by the error line on top of each bar, which show the variation within classifier from different data sets. From the figure, it is clear that there is no significant difference between the classifiers, as most of the 95% confidence intervals for different classifiers overlap. It is apparent that with increasing CM, there is a compromise between the sensitivity and specificity values. If we want a higher sensitivity value, the corresponding specificity value will suffer accordingly. This may imply that

we are seriously compromising the accuracy of the majority class, especially for the data sets where CM > 50, where we are losing accuracy of the majority class of around 29% for a gain in minority class accuracy of around 48%. Moreover in this situation sensitivity is now greater then specificity, which shows that extreme bias in the classifier is introduced in favor of minority class, hence these situations warrant a better re-sampling technique.

For the active sampling (informed sampling techniques), NCL provided the best sensitivity values among all the Neighborhood Cleansing Techniques (NCT) as described in section 4.2.2. In these cases we can see that the accuracy of the majority does not suffer as much in comparison with the US techniques in various CM groups, but the accuracy of minority is lower. For group CM > 50, we found that the decrease in specificity is more in comparison to RUS technique considered in this group, whereas the increase in minority class sensitivity is also less. This again strengthens the inference that with increased overlap (as reflected by CM), improving the accuracy of the minority can seriously compromise majority class accuracy. Hence better re-sampling techniques are needed in these cases.

Among the classifiers, we can see from Fig. 4.4 that SVM performs the best on the bases of high sensitivity, GM and AUC values, when classes are relatively balanced by under sampling, whereas CT and NN prove to be more robust with increasing CM. NN outperforms the other classifiers especially for cases CM > 50, which again strengthens the conclusion that NN is the most robust out of all the classifiers evaluated.

These results concur with similar findings in the studies conducted by Hulse et al. [1] and Jo and Japkowicz [54] for the severe imbalance problem. From the above we may conclude that with increasing complexity (CM), all the base classifiers prefer a balanced class distribution for the training set. This supports recent studies preferring balanced class distribution for the training sets [17, 19, 20, 21].

Ensemble models (bagging, boosting and Random Forest) do better than the base classifier, supporting authors [17, 19] who prefer boosting. EasyEnsembles [17] combines the models by summing the posterior probabilities of all models and normalized to achieve the range from 0 to 1; this was the approach

applied in our studies. We are using the boosting classification technique for the clustering based techniques to give a fair comparison with the repetitive models, with posterior probabilities of all models combined to make decisions. Table 4.7 presents the average evaluation measures for all the ensemble methodologies considered in the study. In addition to Bagging, Boosting, EasyEnsemble and RUSBoost, Clus(bal) [21] is used to select a balanced class distribution from the majority class cluster, and Clus(ind) as proposed by Wu et al. [20]. In all the repetitive models in Table 4.7, accuracy of the minority class is better than the majority class. This is due to the fact that all these techniques are using balanced data for their training sets, except the Clus(ind) [20], which mainly depends on k-mean clustering. We use 'k' (i.e., k=$\frac{N_{maj}}{N_{min}}$) as proposed by Wu et al. [20] to roughly balance the classes, and obviously it does not produce reliable results, hence a different selection of 'k' should be considered to get reasonable models. The results are illustrated in Figure. 4.5. The relative comparison can be summarized as below.

- Easy Ensemble, RUSBoost, and CLUS(bal), are producing almost identical results for the groups with CM < 50, where accuracy of the minority class is slightly better than the majority class.
- For group CM > 50 the results for EasyEnsemble and RUSBoost are better than CLUS (bal). Overall these techniques makes classifier biased in favor of the minority class, and as a result specificity is far less than sensitivity.
- Clus(ind) proposed by [20] should be avoided in all cases, or selection of 'k' in k-mean clustering should be handled with extreme care. This technique depends on k, which can't be determine easily.
- Overall EasyEnsemble and RUSBoost work better than clustering based techniques and this finding can be supported by the study conducted by Molinara et. al. [55], where their results showed that random splitting is superior to the cluster based approach.

From the above results it shows that if we have CM < 50, repetitive under-sampling is resulting in a satisfactory model with Sensitivity and Specificity

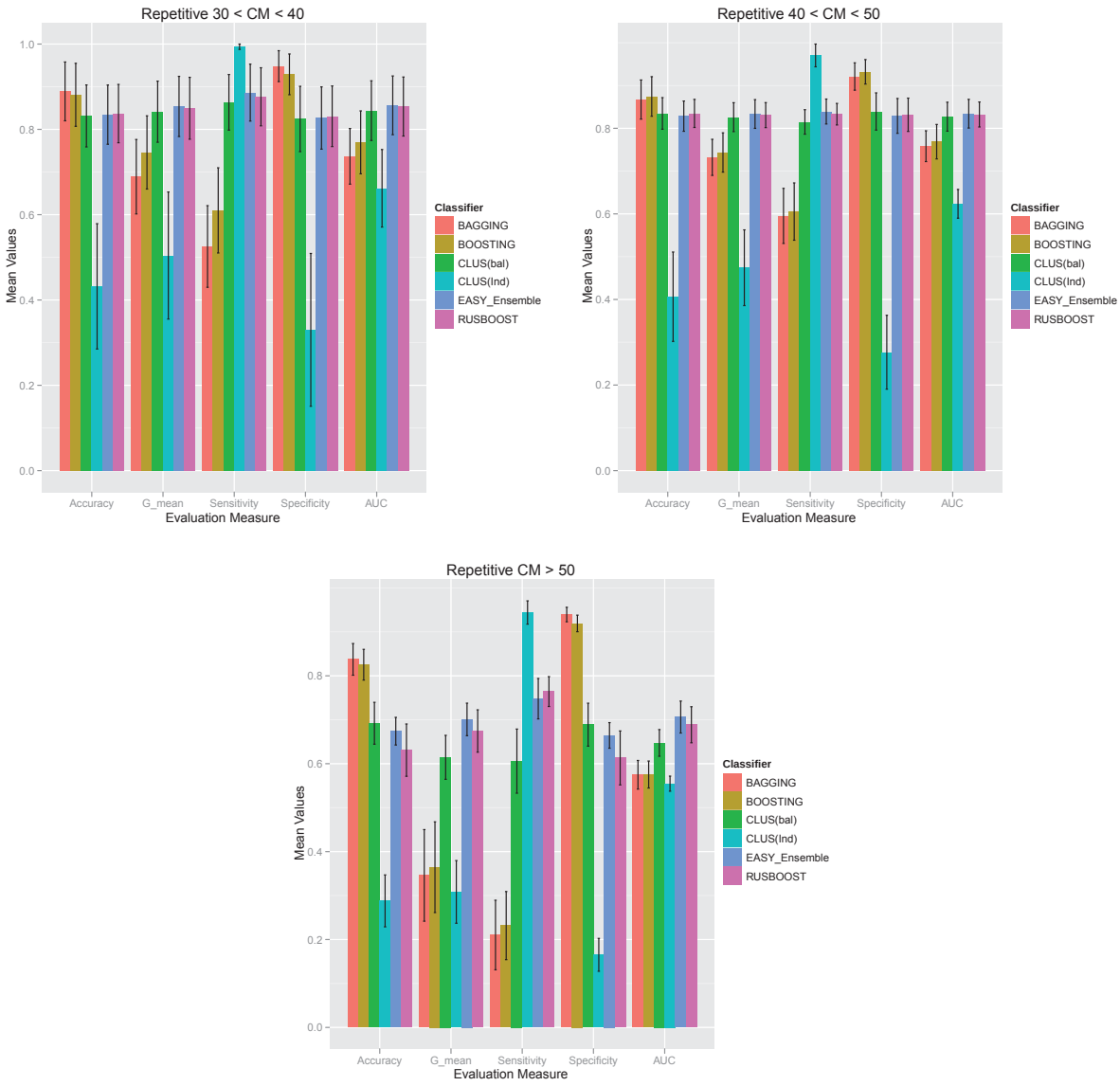

Figure 4.5: A summary of the results obtained by the Repetitive undersampling on the different categories of problems, top left represent Complexity Measurement 0%<CM≤40%, top right 40%<CM≤50% and bottom centered is CM > 50%. The error bar at the top of each bar represent 95% confidence interval. Every Methodology and Evaluation Measure is represented by its names

both greater than 80%. But when CM > 50, this method is not doing well, by severely under-estimating the majority class. Better sampling techniques, perhaps a combination of over-sampling and under-sampling, need to be considered in this situation.

4.4 Discussion

From the above results a number of conclusions can be drawn. We use CM as the complexity measure to describe the difficulty of class learning, in comparison to complexity measurement proposed by Ho [26]. In previous studies the class imbalance ratio (IR) was used as the threshold [1, 3], to describe the level of difficulty between different datasets. But our experiments prove that IR is not an adequate threshold variable to describe the effectiveness of any preprocessing techniques. For example in our second group 30%<CM≤40%, a number of data sets have minority class just under 10%, e.g., Yeast4 (minority class just 3% of the whole data set) and Sat (minority class 9.4%). Nevertheless accuracy of the minority class is relatively better than that of the larger minority class. In contrast, the consistency and uniform behavior of base classifiers within each CM group proves that this may be an appropriate grouping criterion.

For CM $\leq$ 20, there is no need of any preprocessing of data sets, since every classifier considered in the experiment was able to predict classes adequately, with ensemble models performing relatively better than the base classifiers. For 30< CM $\leq$ 50, all the classifiers prefer a balanced class distribution, in comparison to the other under-sampling ratios considered. This is due to the fact that with class imbalance and increased overlap between the classes, the classifier is overwhelmed by the majority class, hence we have high accuracy for the majority class at the expense of minority class. Preprocessing in this group produces acceptable results for the minority class (i.e., accuracy of minority class > 80%) without losing much accuracy for the majority class. For CM> 50, which needs much greater consideration, since techniques that worked for lower CM values were not effective in this group. The balancing act of training data sets severely under-sampled the

majority class, which resulted in losing its accuracy by around 30% and resulted in specificity lower then sensitivity. For this CM region we recommend a combination of over-sampling and under-sampling techniques, which may help in gaining better accuracy for both the majority and minority class. In chapter 5 we have shown that in the overlapped regions over-sampling works better than just under-sampling. A combination of over-sampling and under-sampling potentially creates ideal classifiers [8, 9].

One of the most important conclusions that can be drawn is the failure of the active sampling techniques, like CNN, OSS, WE and Cluster based under-sampling. While NCL and Tomek link produce comparatively better results, these techniques should be used only after careful examination of the data sets, since for example with highly overlapped and severely imbalanced data sets, these techniques will result in severe under-sampling of the minority class, as can be seen from our results. While these techniques seem to be promising in class imbalance situations, we find that simpler techniques like RUS perform better, and these can be used more effectively in the repetitive under-sampling techniques, like EasyEnsemble and RUSBoost tecniques, which produce much better results in comparison to any other technique considered in this study.

4.4.1 Validity of Results:

Here we again refer to the study conducted by the Hulse et al. [1], where they mentioned two types of threats to the validity of the results, namely, internal and external threats. Internal threats refer to unaccounted influences that may impact on the results, whereas external threats refer to the generalization of results outside of the experimental setting.

As mentioned earlier all experiments have been conducted using freely available software in the R system (R Development Core Team, 2004) [35] which also provides the user with a powerful statistical programming language. For different classifiers R packages have been used which are frequently used in the machine learning and data mining community supporting the internal validity of our results.

External validity refers to the reliability and generalization of experimental results. We used 18 UCI data sets in our experiments containing different degrees of imbalance and different sizes, thus providing a diverse test bed. Our results can be compare with other studies conducted by various researchers, [1, 3, 19], in terms of UCI data sets and creating imbalances within a data set, different evaluation values likes sensitivity, specificity and GM and general conclusions about preprocessing techniques.

For random under-sampling we considered a number of partitions, and used Learning Curve (LC) to find the optimal class distribution for training; our work showed that with increased complexity and class imbalance, classifiers prefer the balanced distribution, which was also found by Hulse et al., [1] for severe Imbalance Ratio (IR). The consistency of our results suggest that they are generalizable for implementation outside the range of the experiments, as there is generally the same behaviour among the among the classifiers within different CM groups, making the results more reliable. There is no attempt made to make a particular technique optimal, as OSS and Cluster based under-sampling described how to remove majority class examples; we followed methodology of authors presented in papers, so there was no ability to alter the level of sampling. Hence comparisons with other techniques are reasonable.

4.5 Conclusion

In this work we have analyzed the effect of under-sampling techniques for imbalanced data sets by means of a data complexity measure (CM). We used four base classifiers, three ensemble classifiers with 18 UCI data sets containing different degrees of imbalance and of different sizes, thus providing a diverse test bed. The objective of the research is to compare the methodologies of under-sampling techniques in the existing imbalance literature, and to provide guidance to data miners/researchers building models with imbalanced data sets. From the experiment it is very clear that preprocessing is very important in improving classifier performance, especially when measured by Sensitivity and Geometric Mean. Using the CM as threshold we notice that

most classifiers are behaving very similarly with different under-sampling techniques, which shows the effectiveness of preprocessing techniques. This suggests that we don't have to use different classifiers for the imbalanced data sets, although an ensemble classifier can be used along with preprocessing to improve the result. Since RUS gives better results compared to other more complicated techniques considered in this study, we recommend repetitive under-sampling as a good choice, for least loss of information due to under-sampling of the majority class. In future work, we will explore more and larger data sets, especially for more the complex CM region i.e, CM > 50, where both under-sampling and over-sampling may be advantageous.

References

[1] Hulse, J.V., Khoshgoftaar, T.M., Napolitano, A., Experimental perspectives on learning from imbalanced data, in: Proceedings of the 24th International Conference on Machine Learning, Corvalis, Oregon, pp. 935-942, 2007.

[2] Estabrooks, A., Jo, T. and Japkowicz, N., A Multitple Resampling Method For Learning From Imabalanced Data Sets. Computational Intelligence, Vol. 20, no. 1, pp. 18 - 36, 2004.

[3] Batista,G.E., Prati, R.C., and Monard,M.C., Balancing strategies and class overlapping. In proceeding of 6th International Symposium on Intelligent Data Analysis, Madrid, Spain, pp. 24-35, 2005.

[4] Garcia, V., Mollineda, R., Sanchez, J., S., On the k-NN performance in challenging scenario of imbalance and overlapping. Pattern Analysis Application, Vol. 11, no. 3-4, pp. 269-280, 2008.

[5] Japkowicz, N., and Stephen, N., The class imbalance problem: a systematic study, Intelligent Data Analysis, Vol. 6, no. 5, pp. 429-449, 2002.

[6] Drummond, C. and Holte, R., Severe Class Imbalance: Why Better Algorithms Arent the Answer. In Proceedings of the 16th European Conference on Machine Learning (ECML/PKDD'05), pp. 539-546, 2005.

[7] Weiss, G. and Provost, F., Learning when training data are costly: The effect of class distribution on tree induction. Joural of Artificial Intelligence Research, pp. 315-354, 2003.

[8] Chawla, N., Bowyer, K.,Hall, K., Kegelmeyer, P., SMOTE: Synthetic Minority Over-Sampling Technique, Journal of Artificial Intelligence Research, Vol.16, pp. 321-357, 2002.

[9] Provost, F., Jensen, D., and Oates, T. Efficint Progressive Sampling. Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, San Diego, California, United States, pp. 23-32, 1999.

[10] Iyengar, V. S., Apte, C., AND Zhang, T. Active learning using adaptive resampling. In KDD 00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (New York, NY, USA), ACM Press, pp. 91-98, 2000.

[11] Hulse, J.V., Khoshgoftaar, T.M., and Napolitano, A., An Empirical Comparison of Repetitive Undersampling Techniques, International Conference on Information Reuse and Integration, IRI '09. IEEE, pp. 29-34, 2009.

[12] Tomek, I., Two Modifications of CNN. IEEE Transactions on Systems Man and Communications SMC-6, pp. 769-772, 1976.

[13] Hart, P. E., The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory IT-14, pp. 515-516, 1968.

[14] Kubat, M., and Matwin, S. Addressing the Curse of Imbalanced Data Sets: One Sided Sampling. In the Proceedings of the Fourteenth International Conference on Machine Learning, Nashville, TN, pp. 179-186, 1997.

[15] Laurikkala, J., Improving Identification of Difficult Small Classes by Balancing Class Distribution. Technical Report A-2001-2, University of Tampere, 2001.

[16] Wilson, D. L., Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Transactions on Systems, Management, and Communications, ; Vol. 2, no. 3, pp. 408-421, 1972.

[17] Liu, XY., Wu, J. and Zhou, Z,H., Exploratory under-sampling for class imbalance learnings, in proceeding of the sixth International conference on Data Mining. Hong Kong, China: IEEE Computer Society, pp. 965-969, 2006.

[18] Freund, Y. and Schapire, R., Experiments with a new boosting algorithm, in proceeding of the Thirteenth Intenational Conference on Machine Learning, pp. 148-156, 1996.

[19] Seiffert, C., Khooshgortaar, T. M., Hulse, Van, J. adn Napolitano, A., RUSBoost: Improving classification when training data is skewd, the 19th IEEE International Conference on Pattern Recognition, Tampa, Florida, USA, pp. 1-4, 2008.

[20] Wu, J., Xiong, H., Chen, J., COG: local decomposition for rare class analysis, Data Mining and Knowledge Discovery, Springer Netherlands, Vol. 20, no. 2, pp. 191-220, 2010.

[21] Zhang, YP., Zhang, LN, Wang, YC, Cluster-based majority under-sampling approaches for class imbalance learning," IEEE International Conference on Information and Financial Engineering (ICIFE), pp.400-404, 2010.

[22] Bradley, P. S. and Fayyad, U. M., Refining initial points for K-means clustering, Proceedings of the Fifteenth International Conference on Machine Learning (ICML98), pp. 91-99, 1998.

[23] Barandela, R., Valdovinos RM., Snchez JS, Ferri FJ. The Imbalanced Training Sample Problem: Under or Over Sampling. In Structural, Syntactic, and Statistical Pattern Recognition, pp. 806-814, 2004.

[24] Batista,G.E., Prati, R.C., and Monard,M.C., A study of the behavior of several methods for balancing machine learning training data. IACM SIGKDD Explorations Newsletter, Vol. 6, no. 1, pp. 20-29, 2004.

[25] UCI KDD archive.http://kdd.ics.uci.edu/databases/covertype/covertype.html. 2005.

[26] Ho, T. K., A Data Complexity Analysis of Comparative Advantages of Decision Forest Constructors, In Pattern Analysis and Application, pp. 102-112, 2002.

[27] Hosmer, D. W. and Lemeshow, S., Applied Logistic Regression, Wiley, NewYork:Wiley, 1989.

[28] Morgan, J.and Sonquist, J., Problems in the analysis of survey data, and a proposal. Journal of American Statistical Society, Vol. 58, pp.415-434, 1963.

[29] Quinlan, J. R., C4.5: Programs For Machine Learning. Morgan Kaufmann, San Mateo, California, 1993.

[30] Hertz, J., Krogh, A., and Palmer, R., Introduction to the Theory of Neural Computation, Addison Wesley, Reading, 1991.

[31] Bishop, C.M., Neural Networks for Pattern Recognition, Oxford University Press, NewYork, 1995.

[32] Cheng, B. and Titterington, D.M., Neural Networks: a review from a statistical perspective, Statistical Science, Vol. 9, no. 1, pp.2-54, 1994.

[33] Cristianini, N. and J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, 2000.

[34] Burges, C., A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, Vol. 2, no. 2, pp.1-47, 1998.

[35] R Development Core Team, R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, http://www.r-project.org, 2004.

[36] Ihaka, R. and Gentleman, R., R a language for data analysis and graphics, Journal of Computational and Graphical Statistics, Vol. 5, pp. 299-314, 1996.

[37] Breiman, L., Random Forest, Machine Learning, Vol. 45, no. 1, pp. 5-32, 2001.

[38] Harting, J., Clustering Algorithms, Wiley-Interscience, New-York, 1975.

[39] Kaufman, L. and and Rousseeuw, P.J. Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York, 1990.

[40] Kubat, M., and Matwin, S. Addressing the Curse of Imbalanced Data Sets: One Sided Sampling. In the Proceedings of the Fourteenth International Conference on Machine Learning, Nashville, pp. 179-186, 1997.

[41] Egan, J. P.,Signal detection theory and ROC analysis. Academic Press, London, 1975.

[42] Therneau, T.M., Atkinson B., and R port by Brian Ripley, rpart: Recursive Partitioning. R package version 3.1-42, (http://www.mayo.edu/biostatistics), 2004.

[43] Ripley, B., class: Function for Classification. R package version 7.3-42, http://www.stat.ox.ac.uk/pub/MASS4/, 2010.

[44] Meyer, D., Support Vector Machines. The Interface to libsvm in package e1071. http://www.r-project.org), 2004.

[45] Esteban Alfaro Cortes, Matias Gamez Martinez and Noelia Garcia Rubio, R Package "adabag", 2011.

[46] Friedman, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association, Vol. 32, pp. 675-701, 1937.

[47] Friedman, M. A comparison of alternative tests of significance for the problem of m rankings. Annals of Mathematical Statistics, Vol. 11, pp. 86-92, 1940.

[48] Zar, J. H. Biostatistical Analysis (4th Edition). Prentice Hall, Englewood Clifs, New Jersey, 1998.

[49] Sheskin, D. J. Handbook of parametric and nonparametric statistical procedures. Chapman & Hall/CRC, 2000.

[50] Iman, R. L. and Davenport, J. M. Approximations of the critical region of the Friedman statistic. Communications in Statistics, pp. 571-595, 1980.

[51] Kuncheva, L. and Whitaker, C., Measures of diversity in classifier ensembles, Machine Learning, Vol. 51, pp. 181-207, 2003

[52] Sollich, P. and Krogh, A., Learning with ensembles: How overfitting can be useful, Advances in Neural Information Processing Systems, Vol. 8, pp. 190-196, 1996.

[53] Brown, G. and Wyatt, J. and Harris, R. and Yao, X., Diversity creation methods: a survey and categorisation., Information Fusion, Vol. 6, no. 1, pp. 5-20, 2005.

[54] Jo, T., and Japkowicz, N. Class imbalances versus small disjuncts. *SIGKDD Explorations.*, Vol. 6, pp. 429-450, 2004.

[55] Molinara, M., Ricamato, T. and Tortorella, F., Facing imbalanced classes through aggregation of classifiers, in Proceeding of the 14th International Conference of Image Analysis and Processing, Modena, Italy, IEEE Computer Society, pp. 43-48, 2007.

[56] Ho, T. K., Basu. M, Complexity measures of a supervised classification problems, IEEE Trans Pattern Analysis, Machine learning, Vol. 24, no. 3, pp. 289-300, 2002.

4.6 Appendix

4.6.1 R codes

```
##########################################################
# Collection of Examples of the different algorithms that are
    available to build classification models in R.
#
# includes:
#
# Logistic Regression
# Neural Network
# Support Vector Machine
# Decision Tree
# Random Forests
# Adaptive Boosting and Bagging
##########################################################
# Install relevant packages
##########################################################
library(e1071)
library(rpart)
library(nnet)
library(randomForest)
library(fields)
library(adabag)
library(QuantPsyc)

##########################################################
# Confusion Matrix (Be Careful for the interchanging cell number
    for sensitivity and specificity for different data sets)
##########################################################
conf.mat <- function(true, new)
{
  t <- table(true, new)
  s <- sum(t)
  spe <- t[1,1]/sum(t[1,])
  if (t[2,1]/sum(t[2,])==1) {sen<-0} else {sen <- t[2,2]/sum(t
      [2,])}
  g<-sqrt(spe*sen)
```

```
  acc <- round((sum(diag(t)))/s, 2)
  auc<-(1+sen-(1-spe))/2
  res <- list(conf = t, accuracy = acc, specificity=spe,
      sensitivity=sen,  G_mean=g, AUC=auc)
}

#########################################################
# 1. SETUP DATA
#########################################################

#clear worksace
rm(list = ls(all = TRUE))

#Data considered
wdbc <- read.csv('D:/Class imbalance/Data Sets/wdbc data.R',
    header=F) #define pathway of your data sets
colnames(wdbc)<-c("ID","Diagnosis","X1","X2","X3","X4","X5","X6"
    ,"X7","X8",
  "X9","X10","X11","X12","X13","X14","X15","X16","X17","X18","
      X19","X20",
  "X21","X22","X23","X24","X25","X26","X27","X28","X29","X30")

wdbc = wdbc[,-1]
head(wdbc)
summary(wdbc)
table(wdbc$Diagnosis)

# To make Data Standardized...
wdbcstd<-Make.Z(wdbc[,-1])
wdbcstd<-data.frame(wdbcstd)
wdbc<-cbind(wdbc[,1],wdbcstd)
colnames(wdbc)[1]<-"Diagnosis"
wdbc<-data.frame(wdbc)


#Classes
wdbcmin <- subset(wdbc,wdbc$Diagnosis=="M")
wdbcmaj <- subset(wdbc,wdbc$Diagnosis=="B")
wdbcmin$Diagnosis <- factor(wdbcmin$Diagnosis)
```

```
wdbcmaj$Diagnosis <- factor(wdbcmaj$Diagnosis)
dim(wdbcmin); dim(wdbcmaj)
head(wdbcmin); head(wdbcmaj)

#let introducing common names
dd<-wdbc # for the whole data set
colnames(dd)[30]<-"Class" #change the class name accordingly
ddmin<-wdbcmin
colnames(ddmin)[30]<-"Class" #change the class name accordingly
ddmaj<-ddmaj
colnames(ddmaj)[30]<-"Class" #change the class name accordingly


#Test:Train data (5-fold cv)
set.seed(1234)
n <- nrow(ddmin)
taille <- n%/%5
alea <- runif(n)
rang<-rank(alea)
cvid <- (rang-1) %/% taille + 1
cvid<-as.factor(cvid)
print(summary(cvid))
ddmin = cbind(ddmin,cvid) #minority data with cvids
set.seed(12345)
n <- nrow(ddmaj)
taille <- n%/%5
alea <- runif(n)
rang<-rank(alea)
cvid <- (rang-1) %/% taille + 1
cvid<-as.factor(cvid)
ddmaj = cbind(ddmaj,cvid) #majority data with cvids
head(ddmin); head(ddmaj)
table(ddmin$cvid); table(ddmaj$cvid)


#########################################################
# 2. set the formula
#########################################################
theTarget <- "Class"
```

```
#theFormula <- as.formula(paste("as.factor(",theTarget, ") ~ .
    "))
theFormula <- as.formula(paste(theTarget," ~ . "))
library(caTools) #requireed for AUC calc

#########################################################
# 3. Now just apply the algorithms
#########################################################
#Raw data analyses
clsmtxCTraw = NULL
clsmtxNNraw = NULL
clsmtxRFraw = NULL
clsmtxSVMraw=NULL
clsmtxLRraw = NULL
clsmtxBAGraw = NULL
clsmtxBOOraw = NULL


#Loop for Cross-Validation
for (i in 1:5) {
  ddmintst = subset(ddmin, cvid==i,select=1:ncol(dd))
  ddmintrn = subset(ddmin, cvid!=i,select=1:ncol(dd))
  ddmajtst = subset(ddmaj, cvid==i,select=1:ncol(dd))
  ddmajtrn = subset(ddmaj, cvid!=i,select=1:ncol(dd))
  testset = rbind(ddmajtst,ddmintst)
  trainset = rbind(ddmajtrn,ddmintrn)

LOGISTIC_model <- glm(theFormula, family=binomial(link="logit"),
    data=trainset)
Neural_model<- nnet(theFormula, data= trainset, size=3, decay
    =0.01, maxit=1000)
  #try size=2,4,9,18 and decay=0.001,0.01,0.1,1
SVM_model <- svm(theFormula, data=trainset,type='C',kernel='
    radial',probability = TRUE)
TREE_model <- rpart(theFormula, data=trainset, method="class")
minsplt=round((nrow(ddmintrn))*0.05)
CT_model <- rpart(theFormula,data=trainset, method="class",
    control=rpart.control(minsplit=minsplt, cp=0.0001))
```

```
FOREST_model <- randomForest(theFormula, data=trainset, ntree
    =500)
dd.log<- conf.mat(testset$Class,round(predict(LOGISTIC_model,
    type="response", testset)))
dd.nn<- conf.mat(testset$Class,predict(Neural_model, testset,
    type="class"))
dd.svm<- conf.mat(testset$Class,predict(SVM_model, testset,
    probability = TRUE))
dd.conf<- conf.mat(testset$Class,predict(CT_model, testset, type
    ="class"))
dd.RF<- conf.mat(testset$Class,predict(FOREST_model, testset,
    type="class"))
dd.bg <- bagging(theFormula,data=trainset,mfinal=10,control=
    rpart.control(minsplit=minsplt, cp=0.0001))
dd.adaboost <- adaboost.M1(theFormula,data=trainset, boos=TRUE,
    coeflearn='Breiman', mfinal=10,control=rpart.control(minsplit
    =minsplt, cp=0.0001))
dd.bag<- conf.mat(testset$Class,predict.bagging(dd.bg, testset)$
    class)
dd.boo <- conf.mat(testset$Class,predict.boosting(dd.adaboost,
    testset)$class)



clsmtxLR<-cbind(dd.log$acc,dd.log$G_mean,dd.log$sensitivity,dd.
    log$specificity,dd.log$AUC)
clsmtxNN<-cbind(dd.nn$acc,dd.nn$G_mean,dd.nn$sensitivity,dd.nn$
    specificity,dd.nn$AUC)
clsmtxSVM<-cbind(dd.svm$acc,dd.svm$G_mean,dd.svm$sensitivity,dd.
    svm$specificity,dd.svm$AUC)
clsmtxCT<-cbind(dd.conf$acc,dd.conf$G_mean,dd.conf$sensitivity,
    dd.conf$specificity,dd.conf$AUC)
clsmtxRF<-cbind(dd.RF$acc,dd.RF$G_mean,dd.RF$sensitivity,dd.RF$
    specificity,dd.RF$AUC)
clsmtxBAG<-cbind(dd.bag$acc,dd.bag$G_mean,dd.bag$sensitivity,dd.
    bag$specificity,dd.bag$AUC)
clsmtxBOO<-cbind(dd.boo$acc,dd.boo$G_mean,dd.boo$sensitivity,dd.
    boo$specificity,dd.boo$AUC)
```

```
161
clsmtxLRraw<-rbind(clsmtxLRraw,clsmtxLR)
163 clsmtxNNraw<-rbind(clsmtxNNraw,clsmtxNN)
clsmtxSVMraw<-rbind(clsmtxSVMraw,clsmtxSVM)
165 clsmtxCTraw<-rbind(clsmtxCTraw,clsmtxCT)
clsmtxRFraw<-rbind(clsmtxRFraw,clsmtxRF)
167 clsmtxBAGraw<-rbind(clsmtxBAGraw,clsmtxBAG)
clsmtxBOOraw<-rbind(clsmtxBOOraw,clsmtxBOO)
169     }
colnames(clsmtxCTraw)<-c("Accuracy","G_mean","Sensitivity","
    Specificity","AUC")
171 rownames(clsmtxCTraw)<-rep("CT",5)
colnames(clsmtxRFraw)<-c("Accuracy","G_mean","Sensitivity","
    Specificity","AUC")
173 rownames(clsmtxRFraw)<-rep("RF",5)
colnames(clsmtxNNraw)<-c("Accuracy","G_mean","Sensitivity","
    Specificity","AUC")
175 rownames(clsmtxNNraw)<-rep("NN",5)
colnames(clsmtxLRraw)<-c("Accuracy","G_mean","Sensitivity","
    Specificity","AUC")
177 rownames(clsmtxLRraw)<-rep("LR",5)
colnames(clsmtxSVMraw)<-c("Accuracy","G_mean","Sensitivity","
    Specificity","AUC")
179 rownames(clsmtxSVMraw)<-rep("SVM",5)
colnames(clsmtxBAGraw)<-c("Accuracy","G_mean","Sensitivity","
    Specificity","AUC")
181 colnames(clsmtxBOOraw)<-c("Accuracy","G_mean","Sensitivity","
    Specificity","AUC")

183 clsmtxcombine<-rbind(clsmtxCTraw,clsmtxRFraw,clsmtxSVMraw,
    clsmtxNNraw,clsmtxLRraw,clsmtxBAGraw,clsmtxBOOraw)
display_results()
185 library(xlsx) # to export the results directly into the Excel
    file
write.xlsx(clsmtxcombine,file="D:/Class imbalance/Data Sets/
    Classification results of WDBC..xlsx",sheetName="WDBC",row.
    names=FALSE)
```

Chapter 5

Complexity Measure: A Systematic Approach to Over-sampling of Minority Class in Imbalanced Data Sets

Traditional classification algorithms can be inadequate in their performance on extremely imbalanced data. In this chapter we use a complexity measure as a tool for over-sampling of the minority class examples, which can determine the amount of over-sampling in minority class and fix the parameters for the Synthetic Minority Over-sampling Technique (SMOTE). Experiments show that our approach achieves better TP (true positive: minority class accuracy) and G-mean than original SMOTE, Border-Line SMOTE and random over-sampling methods.

5.1 Introduction

Complexity Measure (CM) can be used in two ways: to measure the level of difficulty in imbalanced data sets and as a tool to help cater for the problem. This chapter will introduce how CM can be used to devise a structured study for learning about class imbalance problems. To date, no researcher has

used complexity of a data set as a measure to devise a systematic strategy to overcome the class imbalance problem. Some authors [1, 2, 3, 4], have considered complexity measurement but never as a way to cater for imbalance rather to indicate the level of difficulty in data sets. In this chapter, we propose a systematic approach for over-sampling which will adjust to different data sets according to their complexity. Data complexity in our context is the quantification of the degree of difficulty in class learning from a given data set.

The underlying difficulty associated with highly imbalanced classification problems suggests that an automatic classification system, working entirely on its own, may not be a realistic goal in the near future. Rather, we feel that a two step system will work better, with an automatic classification system to measure complexity (pre-screening tool) followed by the re-sampling procedure. The first step of classification is to find the geometric location of the minority class. The aim is to make the execution of the more expensive re-sampling procedure (that is the second step) affordable as the pre-screening step effectively narrows down the minority class examples that need over-sampling. The specific goal of this chapter is to present a complexity measurement to be used in the pre-screening step. We develop our approach through the well known k-nearest neighbor (kNN) approach.

The Bayes error provides the lower error bound that can be achieved by any pattern classifier [5, 6]. This error rate will be greater than zero whenever class distributions overlap. When all the class priors and conditional likelihoods are completely known, in theory the Bayes rate can be obtainable [5]. In spite of this importance, Bayes error has not been used directly in class imbalance situations. However when the pattern distributions are unknown, the Bayes error cannot be readily computed. Thus, one cannot know how much classification error is due to class density overlap and how much is due to limitations of the training data (such as class imbalance) or deficiencies in the classifier. It is therefore important to not only design a good classifier but to have a limit or bound on achievable classification rate for a given data set [7]. Such estimates will help researchers to decide whether to improve the current classifier, to use another classifier on the same data set or to acquire

more data using some re-sampling technique. A method for estimating the Bayes error without requiring the class distributions of the data set is based on nearest neighbors (k-NN) [8]. The Bayes error can be given in terms of the error of the NN classifiers. We chose kNN because of its simplicity and its natural inability to address class imbalance problems (because it adopts a local based estimation procedure that suffers significantly from the scarcity of training samples). In this way it is possible to assess the effectiveness of proposed complexity measurement in very critical conditions. The novelty of the proposed approach can be found in the following.

1. We calculate a complexity measure (CM) for each class, being the proportion of difficult cases belonging to that class, where 'difficult' means having a majority of k nearest neighbors in the opposing class. Other complexity measures used in the literature [2] give a single value that cannot be used in structured over-sampling.

2. We propose a kNN approach for addressing the detection problem in class imbalance classification. For data sets with highly imbalanced class sizes, we advocate the kNN approach as a pre-screening tool, after which each minority class example will be re-sampled in proportion to its k-nearest-neighbor complexity. Then we apply a classification procedure to this data set in order to increase the accuracy on minority class examples.

3. As calculation of CM depends on a distance metric, we use Euclidean distance for continuous data and Random Forest distance for categorical or mixed data.

In order to assess the effectiveness of the proposed approach, we use simulation in addition to 13 UCI data sets [9]. Experimental results confirm that the method presented is capable of increasing both the accuracy and robustness of classification in imbalanced data sets.

This chapter is organized in four sections. In Section 5.2, the proposed re-sampling scheme based on complexity measurement is presented. Section 5.3 describes the data sets used in the experiment and the results obtained

by the proposed approach, as well as some specific issues in implementation. Section 5.4 summarizes our conclusions and gives suggestions for future work.

5.2 Over-sampling using Complexity Measure (OSCM)

Our aim is to develop a classification rule $\hat{C}(x, TS)$ with high accuracy for the minority class, in high class imbalance cases. Here $TS = \{(y_i, x_i), i = 1, 2, , 3, ...N\}$ is the training set of size N, $x_i = (x_{i1}, x_{i2}, x_{i3}, ...x_{ip})$ is the vector of p explanatory variables for the ith observation in TS, and y_i is the class indicator. Since we are studying two-class problems, y_i can takes binary values: when $y_i = 0$, the corresponding case belongs to the majority class and when $y_i = 1$ the corresponding case belongs to the minority class. The classification rule $\hat{C}(x, TS)$ is to predict the class of a future case with explanatory variable x.

In a classification problem, the loss function $L(y, \hat{y})$ gives the cost of misclassification between actual response y and predicted value $\hat{y}$. Since we have a binary class, we use the 0/1 loss function i.e., the loss function has a value one when the predicted value is different from the actual class and zero otherwise. This loss function can be expressed as: $L(y, \hat{y}) = \mathrm{I}(y \neq \hat{y})$ where I(.) is the indicator function. Given the loss function, the prediction error(PE), also the generalization error of the classification rule $\hat{C}(x, TS)$, is defined as [18]:

$$PE(\hat{C}) = E_{OF} E_F [L(Y, \hat{C}(x, TS))] \tag{5.1}$$

where E_F is the expectation over selection of the training set TS, whose member are i.i.d and E_{OF} is the expectation over selection of the test observation (Y, x).

Since we are most interested in the accuracy of the minority class, we divide the prediction error into two parts: the false positive rate (FP) and the true positive rate (TP). FP is the expected rate of misclassification of majority class as minority class examples, and TP the expected rate of

correct detection of minority class examples. According to the definition of the prediction error given in Equation (5.1), the true positive and false positive alarm rate of $\hat{C}(x, TS)$ can be defined as:

$$TP(\hat{C}) = E_{OF}\{E_F[1 - L(Y, \hat{C}(x, TS))]|Y = 1\} \quad (5.2)$$

$$FP(\hat{C}) = E_{OF}\{E_F[L(Y, \hat{C}(x, TS))]|Y = 0\} \quad (5.3)$$

We use a kNN approach to serve as a pre-screening tool, to analyze the geometric situation, especially for the minority class, in the original data set. Re-sampling in specific regions of the minority class will results in enhanced true positive rate for minority class. It is known that increasing the TP of a classifier will generally lead to increase in the FP as well. Thus we need to regulate the FP to be below a specific percentage value while increasing the true positive rate. Since we are using a kNN approach, this process can be modeled as a local estimation of the posterior probabilities of class based on the relative frequency of the class labels in the neighborhood (defined by the k closest training samples). Given the variable x_i, the kNN estimate $\hat{P}(Y_i|x_i)$ of the conditional posterior probability $P(Y_i|x_i)$ is obtained according to the analysis of labels of the k samples (included in the TS) closest to x_i (which define the neighborhood N_i). The classification rule can be written as:

$X_i \in Y$ if and only if

$$Y = \underset{Y_i}{\text{argmax}} \left(\frac{\text{number of patterns} \in Y_i \text{ in } N_i}{k} \right) \quad (5.4)$$

It is worth noting that the kNN technique is mostly unsuitable as a classifier for severe imbalance classification problems. However, in this study, we are just using kNN to construct a complexity measure in order to re-sample the minority class, thus increasing the accuracy for the minority class more efficiently.

5.2.1 Complexity measure

In our approach, we calculate a complexity measure for the minority class only. Calculation involves finding the k nearest neighbors of every data point in the class, where k is odd. If the majority of the neighbors are in the same class, this point is designated as easy to classify; if most of the neighbors are in the opposing class, it is difficult. Complexity can be defined as:

$$CM_k(x_i, Y_i) = \mathrm{I}\left(\frac{\text{number of patterns} \in Y_i = 1 \text{ in } N_i}{k} \leq 0.5\right) \tag{5.5}$$

The overall complexity measure is the proportion of points classified as difficult, but this can be decomposed into separate measures for the complexity of each class. Typically k should be large enough to use this approach properly. We recommend to use an odd number for k to avoid ties in deciding the complexity measure. Deciding on k is a difficult problem. We formulate our k in the kNN approach as:

$$k = \min\{k \geq 3 : (CM_k - CM_{k-2}) \leq \alpha\} \tag{5.6}$$

where α is a suitable small threshold to regulate k (for more detail, see chapter 3, section 3.5.1), which eventually controls the amount of re-sampling of minority class and helps in regulating FP. Simulation experiments from chapter 3 suggest that for low overlap, higher nearest neighbor will give us the same complexity, so we prefer lower k, and for higher overlap the complexity measure will stabilize as k increases. Hence we determine k according to equation 5.6. Solving this optimization problem analytically is not possible, so we solve it using a data driven approach, increasing k until the complexity stabilizes.

5.2.2 SMOTE

Researchers have dealt with class imbalance by over-sampling of minority class or under-sampling of majority class [24, 25]. The effect of over-sampling is to more clearly define regions in the feature space as the decision region of

the minority class [11].

SMOTE (Synthetic Minority Over-sampling Technique) was proposed to counter the effect of having few observation of the minority class in a data set [11]. This technique is operated in the feature space rather than the data space, creating synthetic observations of the minority class.

Since this technique depends on distance matrices, we need to deal with nominal (or discrete) and continuous variables differently in SMOTE. Different distance functions can be used for determining the nearest neighbor computation, the simplest and most common way being to use the heterogeneous Euclidean/overlap metric (HEOM) [14] as proposed in [12]. Tsymbal et. al. [13] show that Random Forest distances are better than the commonly used heterogenous Euclidean/Overlap metric (HEOM) of Wilson and Martinez [14] for the categorical or mixed type data sets. In HEOM, the Euclidean distance is used with numeric features, and the overlap distance with categorical features in order to find a distance between two examples. The Euclidean distance for numeric features and the overlap distance for categorical were demonstrated to be robust and difficult to compete with in many applications, see for example [14]. We use Euclidean distance for the continuous variables, and Random Forest distance [15] for the nominal variables/mixed datasets. The new synthetic minority class examples are created as follows:

For the continuous variables

- Take the difference between a minority class variables vector and one of its k nearest neighbors (minority class);
- Multiply this difference by a random number between 0 and 1;
- Add this difference to the variable value of the original variable vector thus creating a new variable vector.

For the categorical variables

- Take the statistical mode of the variable in consideration over its k nearest neighbors(minority class). In case of tie choose at random.

- Assign that value to the new synthetic minority class.

Using this technique, a new minority class example is created along the line segment joining the minority class and its nearest neighbor. Hence using SMOTE, a more general region is learned for the minority class that allows a classifier to better predict test examples belonging to minority class. A combination of SMOTE and under-sampling creates potentially ideal classifiers [11, 16].

5.2.3 SMOTE using complexity measure: (SCM)

In this chapter we propose an algorithm that combines Synthetic Minority Over-sampling Technique (SMOTE) and the complexity measure. We want to utilize SMOTE for improving the accuracy of the minority class and we want to utilize complexity measure to not sacrifice the accuracy over the entire data set and to be able to adapt to the different natures of data sets. The major goal is to better model the minority class in the data set, by providing the classifier not only minority class examples in the data set that were difficult to learn (values detected by complexity measure), but also broader representation of those observations. We want to improve the overall accuracy of the minority class by focusing on the difficult minority class examples, as we want to model this class better. Our goal is to improve accuracy for the minority class.

In the literature all the over-sampling schemes like simple random over-sampling, SMOTE [11], Borderline SMOTE [17] give equal weights to all minority class examples. We believe not every minority class example needs to be over-sampled equally, but proportional to its complexity (level of difficulty). Han et al. [17] presented a modification of SMOTE technique [11] known as borderline-SMOTE (BSM). BSM selects minority class examples which are considered to be on the border of the minority decision region in the feature-space and only perform SMOTE to over-sample those instances, rather than over-sampling all or a random subset, but once again all the observations which are considered as borderline are over sampled equally using SMOTE. Moreover if none of nearest neighbors of a minority class example

are from the same class, it will be regarded as noise and not over sampled. For more detail see [17]. In fact, if we have a severe imbalanced data set, we will see the majority of the minority class completely surrounded by the majority. If we ignore these examples for over sampling we are not adding much of the information about these examples in the classifier, which will result in a much lower predictive model for these examples and we have to heavily over-sample the rest of the examples to achieve a good predictive model for the minority class.

Since our aim is to reduce the bias against the minority class inherent in the learning procedure due to class imbalance, we need to increase the sampling weights for the minority class. By introducing the SMOTE procedure proportional to the complexity measurement, we are particularly interested in increasing the probability of selection for the difficult minority class cases that are dominated by the majority class points. It is a well known fact that SMOTE has two parameters; number of nearest neighbors (k) and amount of over-sampling (N), usually at the user's discretion. In SMOTE and Border line SMOTE the nearest neighbors (k) is fixed as 5 and a number of iterations are used to decide the optimal level of over-sampling (N). In our approach, once the threshold for the complexity measure CM i.e. α is decided the rest of the procedure will be automatic, i.e., no decision about the number of nearest neighbors and over-sampling is required. Unlike the existing over-sampling methods our methods only over-sample or strengthen those examples which are very difficult to learn. The details of our procedure are outlined below.

First calculate the complexity of each observation for the minority class. For those observations whose nearest neighbor majority is from the other class, synthetic examples are generated in proportion to number of examples from the other class, and added to the original training set. As defined earlier, the whole training data is TS, the minority class in TS_1 and the majority class is TS_0 where

$$TS_1 = \{TS_{11}, TS_{12}, ..., TS_{n_1}\}, \text{ and } \quad TS_0 = \{TS_{01}, TS_{02}, ..., TS_{n_0}\}$$

where n_1 is the size of minority class and n_0 is the size of majority class.

Step 1 For every $TS_{1i} \quad (i = 1, 2, ..., n_1)$ in the minority class TS_1, we calculate its k nearest neighbors in the whole training data TS. The complexity of a minority class data point is measured as:

$$CM_{1i} = \mathrm{I}\left(\frac{\text{number of patterns} \in TS_{1i=1} \text{ in } TS}{k} \leq 0.5\right) \tag{5.7}$$

The number of majority class among the k nearest neighbor is denoted by $\acute{n_0}$.

Step 2 In this step we generate $\acute{n_0}$ minority class examples using SMOTE. For each CM_{1i}, we calculate k nearest neighbors in TS_1. Using SMOTE as define in Section 5.2.2.2 we generate $TS_{1SMOTE} = CM_{1i} * \acute{n_0}$.

Step 3 $TS_{1new} = TS_1 \cup TS_{1SMOTE}$

Step 4 We repeat the above procedure for each CM_{1i}.

It should be kept in mind that we already determine k i.e. number of nearest neighbors using equation 5.6. According to the definition of SMOTE, new synthetic data are generated along the line between the minority class examples and their nearest neighbor from the same class. In another variation, we use simple over-sampling using our complexity measure of minority class, i.e. instead of using SMOTE for difficult minority class examples we use simple over-sampling in proportion to number of majority class examples.

5.2.4 Performance measure for over sampling using SMOTE:

We can quantify the advantage of an over-sampled classifier over its base classifier by two measures [18]: Aggregation effect(AE) and Variance of the base classifier (Var). AE is define as:

$$AE = PE(Y, \hat{C}) - PE(Y, \hat{C}_{CM}) \tag{5.8}$$

From Equation 5.8 we can note that AE is the reduction in prediction error of the classifier using complexity measurement over-sampling $\hat{C_{CM}}$ over the

base classifier $\hat{C}$. As we are dealing with severe imbalance and we are more interested in true positive (TP) and false positive (FP), so the aggregation effect can be decomposed into TP and FP as:

$$AE_{TP} = TP(\hat{C}_{CM}) - TP(\hat{C}) \tag{5.9}$$

$$AE_{FP} = FP(\hat{C}_{CM}) - FP(\hat{C}) \tag{5.10}$$

The variance of the base classifier is define as:

$$\begin{aligned} Var(\hat{C}) &= PE(\hat{C}, \hat{C}_{CM}) && (5.11) \\ &= E_F E_{OF}\{L(x, \hat{C})|Y = 1\} && (5.12) \end{aligned}$$

When using the 0/1 loss function, $Var(\hat{C})$ is the expected rate at which the classifier predicts the class differently from the over sampled classifier using complexity measure. As usual high variation in the predictions will shows instability of the classifier. Section 5.3 and Section 5.4 will illustrate how much prediction improves in UCI data sets [9], when we over-sample the difficult class by using complexity measure instead of randomly over-sampling the whole data set.

5.3 Experiments

As stated earlier Bayes error can be computed in terms of the error of the nearest neighbor (NN) classifiers. In our simulation study we want to investigate with known optimal Bayes error and different class imbalances scenarios: How k-NN will behave in capturing the complexity of the data sets (Bayes error)? Can k be determined? First we investigate with the simulation study and then use these general rules in the real data sets (UCI).

5.3.1 Choice of k

For detailed results of the simulation study, see chapter 3. Fifteen scenarios with Bayes error of (0.4,0.3,0.1) and with imbalances ratio (IR) of

(1,2,2.9,5,11.1) were considered. Here IR is defined as the number of majority class examples divided by the number of minority class examples.

To summarize the findings of this simulation study:

1. k-nn where $k > 2$ may be used in order to capture the complexity of classes with imbalance data;

2. The complexity measures proposed by the Ho and Basu [2] work well for balanced data but cease to capture complexity in imbalance data sets. This is a disadvantage as severe imbalance (minority class $\leq 10\%$) seriously degrades classifier performance;

3. As k increases, k-nn becomes more consistent with the Bayes error;

4. As the imbalance increases our complexity measure for the minority class increases appropriately;

5. The choice of number of nearest neighbors, k, for CM is fixed as defined in Chapter 3, section 3.5.1.

The main purpose of our complexity measure is to see the effect of class imbalance with different levels of Bayes error.

We use the condition as define in equation 5.6, and fix the threshold as (see section 3.5.0.1 for more detail):

$$\alpha = 0.025 \times \text{number of observations in minority class} \tag{5.13}$$

5.3.2 Real Data Examples

To investigate the performance of OSCM with real data, we have used 13 data sets from the UCI data repository [9]. Information about these data sets is summarized in Table 5.1. In order to have the same minority and majority class ratio, stratified 5-fold cross validation was used to obtain training and test subsets (ratio 4:1) for each data set.

To improve the performance of the classification tree (CT)[19] in imbalanced applications, CT was used as a base classifier in all experiments conducted. Furthermore we compared our algorithm with the over-sampling

Table 5.1: Description of UCI data sets

Data sets	♯ of Attributes	Size	Target	$\sharp Min$	$\sharp Max$	$\%Min$
Abalone	8	4177	Ring=7	391	3786	9.4
Abalone	8	4177	Ring<7	448	3728	10.7
Balance	4	625	Balance	49	576	7.8
Car	6	1728	acc	384	1344	22.2
Cmc	9	1473	class 2	333	1140	22.6
Haberman	3	306	class 2	81	225	26.5
Ionosphere	34	351	bad	126	225	35.9
Pima	8	768	class 1	268	500	34.9
Satimage	36	6435	class 4	626	5809	9.7
Vehicle	19	846	opel	212	634	25.1
WDBC	34	569	malignant	212	357	37.3
WPBC	34	198	recur	47	151	23.7
Yeast	8	1484	ME3	163	1321	11

strategy known as Border-line SMOTE (BSM)[17], due to fact that Han et al.[17] have already shown superiority of their technique over SMOTE [11]. Moreover we extended their technique to included categorical variables using Random Forest distances. For all classification tree classifiers, we use pruned trees (see Appendix) in all the runs for each data set. Thus the algorithms could be compared without the influence of CT parameters (they are fixed for all the data sets). For the Border-line Smote algorithm the percentage of minority class over-sampling for each data set is shown in Table 5.2. The parameter k for the complexity measurement (CM) algorithm was set empirically from Equation 5.6, along with resultant amount of over-sampling using Smote or simple random over-sampling. For the categorical or mixed type of data sets we used Random Forest distance matrices for SMOTE and OSCM, and Euclidean distance matrices for data sets with only numerical variables.

All programs were written in the statistical environment R [20], and relevant R codes for our algorithm are provided in Appendix 5.6. For the construction of tree models we used the `RPART` package [21], for nearest-neighbor based on Euclidean distance we used R package `class` [23] with a little modification of the input and output, and to compute Random Forest distances we used the R package `randomForest` [22].

Table 5.2: Characteristics of the 13 data sets we used in experiment: Nature of Variables and Class difference. For some data sets the class label in the parentheses indicate the target class we chose. Moreover, this table shows the choice of optimal over-sampling shown by % Over and relative figure show percentage amount of oversampling in the minority class for Border line Smote and our Complexity Measurement (CM) algorithm with parameter k along with resultant amount of over-sampling

Data sets	Attributes	Class Difference	% Over	k	♯ of Examples Added
Abalone (7)	numerical	3394	700%	7	1931
Abalone (<7)	numerical	3280	400%	7	1148
Balance	Categorical	527	400%	3	133
Car	Categorical	960	300%	3	281
Cmc	Categorical	807	300%	3	468
Haberman	numerical	144	300%	3	61
Ionosphere	numerical	99	100%	3	51
Pima	numerical	233	100%	3	114
Satimage	numerical	5148	400%	5	546
Vehicle	numerical	522	200%	3	274
WDBC	numerical	145	200%	3	40
WPBC	numerical	104	300%	3	93
Yeast	numerical	1158	400%	3	131

After the setting of parameters for CT, CT performance was evaluated using the appropriate metrics for imbalanced classification. For each metric, the mean and standard deviation were calculated from 5 runs with different training and test subsets obtained from stratified 5-fold cross validation. The metrics used in the evaluation process and the average results achieved for the test set are presented in detail in Section 5.3.2.1 below.

5.3.2.1 Results

Table 5.3 illustrates the results using the sensitivity measure (accuracy of minority class). Sensitivity is chosen because we are interested in accuracy for the minority class rather than overall accuracy. Note that in all of the 13 UCI data sets, our algorithm achieved better results than BSM and the original Classification Tree (the best results are marked in bold).

It is worth to note that in the case of the Abalone(7) and Balance data sets both characterized by a huge degree of imbalance, the original Classification Tree was unable to give satisfactory sensitivity values, whereas our

Table 5.3: Sensitivity comparison of methods applied to UCI data sets. Figures gives mean (standard deviation) of sensitivity values calculated using stratified 5 fold cross validation

Data sets	Classification Tree	SMOTE using CM	OS using CM	BSM
Abalone(7)	0.2(0.071)	0.68(0.062)	**0.8(0.026)**	0.67(0.026)
Abalone(<7)	0.59(0.03)	0.73(0.004)	**0.85(0.015)**	0.78(0.029)
Balance	0(0)	0.53(0.103)	**0.78(0.122)**	0.53(0.08)
Car	0.84(0.064)	**0.96(0.036)**	**0.96(0.024)**	0.95(0.017)
CMC	0.29(0.041)	0.66(0.044)	0.62(0.042)	**0.67(0.031)**
Haberman's	0.39(0.043)	0.58(0.042)	**0.66(0.058)**	0.47(0.121)
IonoSphere	0.86(0.041)	0.89(0.041)	**0.91(0.038)**	0.86(0.09)
Pima	0.62(0.107)	0.69(0.0357)	**0.71(0.029)**	0.67(0.035)
SatImage	0.57	**0.65**	0.62	0.63
Vehicle	0.56(0.085)	0.69(0.0184)	**0.78(0.025)**	0.69(0.073)
WDBC	0.88(0.021)	**0.9(0.025)**	**0.9(0.037)**	0.88(0.017)
WPBC	0.44(0.141)	0.76(0.066)	**0.86(0.045)**	0.6(0.184)
Yeast	0.72(0.035)	0.73(0.052)	**0.88(0.058)**	0.71(0.044)

algorithm worked well. Moreover, notice that in all cases over-sampling (OS) using Complexity Measurement is performing very well, in comparison with SMOTE algorithm using Complexity Measurement (SCM) or BSM. If we compare SCM and BSM since both are using SMOTE, out of 13 data sets, 11 times (including two ties) our algorithm performs better than BSM, although all the results are very close. It must be kept in mind that we are not controlling the class distribution of the data: once k is fixed by using the equation 5.6, the rest of the process is automatic. Conversely, SMOTE and BSM need optimization and user discretion to fix the class distribution. As shown in Table 5.2, column (% Over), shows the percentage of over-sampling using SMOTE in minority class in order to get competitive results in comparison to our proposed algorithm. It is known that in imbalanced data, when we want to increase true positive (TP), false positive (FP) also tends to increase, so our aim is to increase the accuracy of the minority class without sacrificing much of the accuracy of majority class.

In Table 5.4 we evaluated the algorithms using the metric G-mean defined as $\sqrt{TP \times TN}$ [10], which corresponds to the geometric mean between the correct classification rates for positive (sensitivity) and negative (specificity) examples, respectively. Hence we want to see how much accuracy of the ma-

jority class suffers due to over-sampling of the minority class. As shown in Table 5.4, over-sampling using the complexity measurement produced better results when compared to original Classification Tree and BSM algorithms. Our algorithm's performance was superior especially for data sets with higher imbalance, which is seen more frequently in real world problems. If we compare SMOTE using our algorithm and BSM, although the results are very close, out of 13 UCI data sets, 9 times our method is better including 3 ties and in 4 data sets BSM is slightly better.

Table 5.4: G-means comparison of methods applied to UCI data sets. Figures gives mean (standard deviation) of sensitivity values calculated using stratified 5 fold cross validation

Data sets	Classification Tree	SMOTE using CM	OS using CM	BSM
Abalone(7)	0.43(0.068)	0.75(0.001)	**0.8(0.0001)**	0.77(0.0002)
Abalone(<7)	0.75(0.016)	0.82(0.002)	**0.87(0.01)**	0.85(0.0002)
Balance	0(0)	0.66(0.051)	**0.82(0.059)**	0.67(0.045)
Car	0.89(0.037)	**0.95(0.0002)**	0.94(0.023)	0.94(0.01)
CMC	0.5(0.034)	**0.72(0.0004)**	0.67(0.007)	**0.72(0.019)**
Haberman's	0.54(0.039)	0.68(0.002)	**0.74(0.029)**	0.60(0.083)
IonoSphere	0.87(0.028)	**0.9(0.026)**	**0.9(0.037)**	0.86(0.05)
Pima	0.68(0.039)	**0.71(0.009)**	0.71(0.023)	0.71(0.028)
SatImage	0.74	**0.76**	**0.76**	0.75
Vehicle	0.69(0.05)	0.74(0.023)	**0.79(0.029)**	0.74(0.038)
WDBC	0.91(0.023)	0.91(0.012)	**0.94(0.012)**	0.92(0.017)
WPBC	0.58(0.085)	0.70(0.004)	**0.79(0.012)**	0.65(0.104)
Yeast	0.83(0.022)	0.83(0.031)	**0.92(0.029)**	0.82(0.023)

Average ROC (Receiver Operating Characteristics) curves [26] were plotted for all data sets and gave similar results. The ROC curve for a binary classifier shows the true positive rate as a function of false positive rate when the decision threshold varies. To illustrate, Fig 5.1 shows an example from the Pima Indian Diabetic data set. Note that our algorithm generates better a ROC curve than BSM and original Classification Tree.

This can be validated from the area under curve (AUROC) which can be obtained from the following equation:

$$AUC = \int_0^1 \frac{TP}{P} d\frac{FP}{N} = \frac{1}{P.N} \int_0^N TP dFP$$

The mean AUC for original classification tree (CT) is 0.74, for Smote using Complexity Measure 0.79 (SCM), BSM 0.77, and over-sampling using

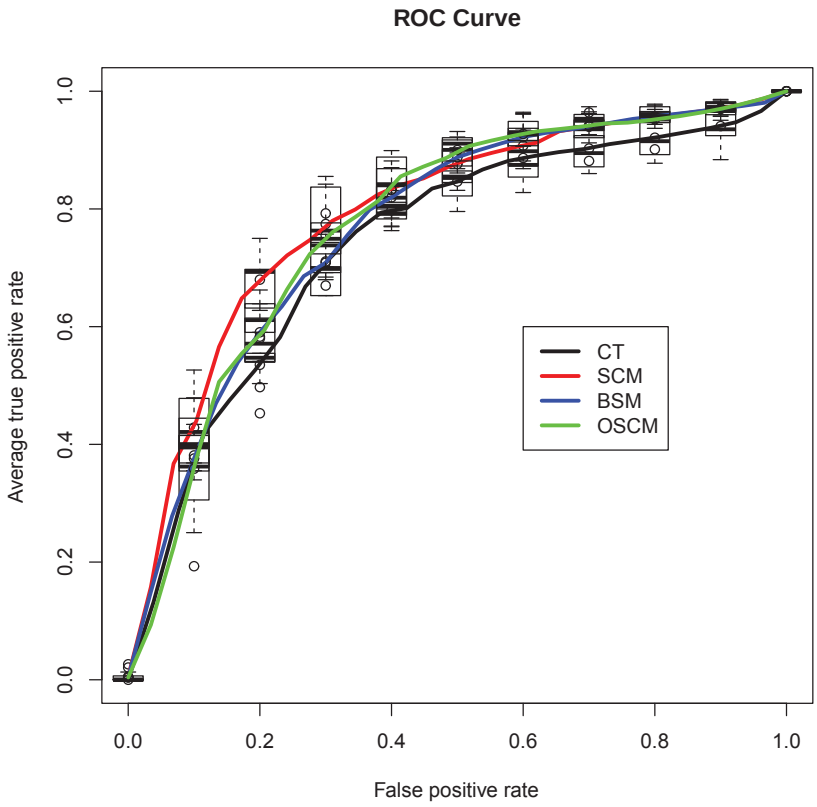

Figure 5.1: Average ROC curve for test set obtained from Pima Indian Diabetic data sets

Complexity Measure (OSCM) 0.77.

5.4 Discussion:

In this section we indicate the advantages of having over-sampling by complexity measure (OSCM), by quantifying the improvement in prediction performance. The aggregation effect and variance of a base classifier, as defined in Equations 5.9 to 5.12, indicate the advantages of OSCM over simply using a base classifier.

The aggregation effect $\widehat{AE(C)}$ can be obtained by taking the difference in prediction of minority class between the base classifier and OSCM classifier. The prediction error for the minority class is given by

$$\widehat{PE(\hat{C}_{TP})} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\sum_{i=1}^{CV} L(Y_i = 1, \hat{C}(x_i, TS_{cv}))}{CV}\right) \tag{5.14}$$

where N is population size of the test set (having the same ratio of minority to majority class as the original data set) and TS_{cv} is the training set. We use a cross validation method for multiple test sets, where we average the prediction errors from individual test sets. Similarly the prediction error for the OSCM is given by

$$\widehat{PE(\hat{C}_{CM(TP)})} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\sum_{i=1}^{CV} L(Y_i = 1, \hat{C}_{CM}(x_i, TS_{cv}))}{CV}\right) \tag{5.15}$$

Then the aggregation effect is:

$$AE_{TP} = PE(\hat{C}_{TP}) - PE(\hat{C}_{CM(TP)}) \tag{5.16}$$

Note that the order of difference of AE between OSCM and the base classifier is changed from the original definition of aggregation effect, to ensure that a positive value of AE indicates improved prediction accuracy over the base classifier. The variance of the base classifier defined in Equation 5.11 is the rate at which the base classifier differs from the OSCM, when using 0/1 loss function. If the resulting variance is high, it indicates that the base classifier is unstable. The variance of base classifier is obtain as

$$\widehat{Var(\hat{C})} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\sum_{i=1}^{CV} L\left(\hat{C}_{CM}(x_i, TS_{cv}), \hat{C}(x_i, TS_{cv})\right)}{CV}\right) \tag{5.17}$$

The variance of True Positive (Var_{TP}) can be obtain in a similar way.

Table 5.5: Over-sampling Effect: True Positive and False Positive of the minority class and Variance of base classifier for True Positive and False Positive

Data sets	AE_{TP}	AE_{FP}	Var_{TP}	Var_{FP}
Abalone(<7)	0.13688844	0.04187922	0.5478618	0.0217451
Haberman	0.1910714	-0.03999998	0.5090195	0.139613
Pima Indian	0.07174774	0.018	0.3604534	0.1323391
SatImage	0.0758294	0.0743432	0.6224784	0.0163339

For illustration we present in Table 5.5 the aggregation effect for True Positive and False Positive of the minority class (column 2 and 3) and variance of base classifier for True Positive and False Positive (columns 4 and 5) for four data sets. The results show a noticeable increase in the detection power for the minority class when over-sampling using complexity measure is used. For example for the Abalone data set OSCM classifies the age less than 7 by 13.7% more than the base classifier, whereas false positives have risen only by 4.2%. For the Haberman data set minority class accuracy has risen by 19.1% and the accuracy of the normal cases has actually increased by 4%. It is obvious that the increase from the over-sampled classifier over the base classifier is more significant in its true positive rate than its false positive rate.

It can be seen from the Table 5.5 that between 36% to 55% (column 3) of predictions are different between the base classifier and the over-sampling classifiers in detection of minority class. The base classifier show a high variation in false positive rate as well. It is reasonable to conclude that the base classifier is not suitable for generating a stable prediction model for these class imbalance data sets.

5.5 Conclusions and Future Work

It is well known that representativeness in the training data is the most important step to have a classifier with high generalization performance. However in most classification problems, representation is not only costly but a very cumbersome task. In general data sets available are small, sparse, with missing values and with highly imbalanced prior probabilities.

In order to address the problem and to bring some structure to over-sampling within the data sets, we have proposed two new methods of over-sampling: synthetic over sampling technique using complexity measure and random over-sampling using complexity measure. Experimental results for several imbalanced data sets has indicated that the proposed algorithm can result in better prediction of minority class, whereas prediction performance of majority class does not suffer much. Data sets used in our experiments contained different degrees of imbalance and different sizes, thus providing a diverse test bed.

Introducing complexity measure (CM) as a pre-screening tool helps us to identify difficult examples belonging to minority class, and simple over-sampling or SMOTE algorithm improves the performance of a classifier on these minority class examples. Therefore, over-sampling these difficult examples forces the classifier to focus more on the difficult examples that belong to minority class rather than to the majority class. By introducing over-sampling proportional to the level of difficulty we were able to dictate the amount of SMOTE or over-sampling, which is still an open question in the data mining community. Unlike SMOTE or BSM we do not need optimization or iterative over-sampling in order to get favorable accuracy of the minority class. Experiment has indicated that our proposal leads to better behavior (high accuracy of minority class), which validates the efficiency of our methods.

In the next chapter we will examine feature selection, and its effect on the complexity measurement.

References

[1] Weng, C. G. and Poon, J., A data complexity anylysis on imbalanced datasets and an alternative imbalance recovering strategy, Proceeding of the IEEE/WIC/ACM International Conference on Web Intelligent, 2006.

[2] Ho, T. K., Basu. M, Complexity measures of a supervised classification problems, IEEE Trans Pattern Analysis, Machine learning, Vol. 24, no.3, pp. 289-300, 2002.

[3] Bernado-Mansilla, E., Ho, T., K., Domain of competence of XCS classifier system in complexity measurement space. IEEE Trans Evol Computation, Vol. 9, no. 1, pp. 82-104, 2005.

[4] Li, Y., Member, S., Dong, M., Kothari, R., Classificability-based omnivariate decision trees, IEEE Trans Neural Netwrok, Vol. 16, no. 6, pp. 1547-1560, 2005.

[5] Fukunga, K., Introduction to Statistical Pattern Recognition, 2nd edition. Boston: Academic Press, 1990.

[6] Devijer, P. A., and Kittler, J., Pattern Recongnition: A Statistical Approach. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[7] Tumer, K., and Ghosh, J., Bayes Error Rate Estimation Using Classifier Ensembles, Smart Engineering System Design, Vol. 5, pp. 95-109, 2003.

[8] Cover, T. M., and Hart P. E., Nearest Neighbor pattern classification. IEEE Transactions on Information Theory, Vol. 13, pp. 21-27, 1967.

[9] UCI KDD archive.http://kdd.ics.uci.edu/databases/covertype/covertype.html, 2005.

[10] Kubat, M., and Matwin, S. Addressing the Curse of Imbalanced Data Sets: One Sided Sampling. In the Proceedings of the Fourteenth International Conference on Machine Learning, Nashville, pp. 179-186, 1997.

[11] Chawla, N., Bowyer, K.,Hall, K., Kegelmeyer, P., SMOTE: Synthetic Minority Over-Sampling Technique, Journal of Artificial Intelligence Research, Vol.16), pp. 321-357, 2002.

[12] Tsymbal A., Puuronen S. Bagging and boosting with dynamic integration of classifiers. In: Zighed, D.A., Komorowski, J., Żytkow, J. (eds.), Principles of Data Mining and Knowledge Discovery, Proceedings of PKDD2000, Springer, LNAI 1910, pp. 116-125, 2000.

[13] Tsymbal, A., Pechenizkiy M., Cunningham P., Dynamic integration with random forests, in: ECML06, 17th European Conference on Machine Learning, Berlin, Germany , pp. 801-808, 2006. (extended version is available online at http://www.cs.tcd.ie/publications/tech-reports/reports.06/TCD-CS-2006-23.pdf).

[14] Wilson D.R., Martinez T.R. Improved heterogeneous distance functions. Journal of Artificial Intelligence Research, Vol. 6, no. 1, pp. 1-34, 1997.

[15] Breiman, L., Random Forest, Machine Learning, Vol. 45, no. 1, pp. 5-32, 2001.

[16] Provost, F. and Fawcett, T., Robust Classification for Imprecise Environment, Machine Learnig, Vol. 42, no. 3, pp. 203-231, 2001.

[17] Han, H., Wang, W., Mao, B., Border-SMOTE: A New Over-Sampling in Imbalanced Data Sets Learning. LNCS Springer, Heidelberg, pp. 878-887, 2005.

[18] Tibshirani, R. (1996) Bias, variance and prediction error for classification rules. Technical report, Statistics Department, University of Toronto, Toornto, Ontario, Canada.

[19] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J., Classifications and Regression Trees, Wadsworth and Brooks, Belmont, 1984.

[20] R Development Core Team, R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, http://www.r-project.org, 2004.

[21] Therneau, T.M., Atkinson B., and R port by Brian Ripley, rpart: Recursive Partitioning. R package version 3.1-42, (http://www.mayo.edu/biostatistics), 2004.

[22] Breiman, L., Random Forest, Machine Learning, Vol. 45, no. 1, pp. 5-32, 2001.

[23] Ripley, B., class: Function for Classification. R package version 7.3-42, http://www.stat.ox.ac.uk/pub/MASS4/, 2010.

[24] Kubat, M. Holte, R. and Matwin, S., Machine Learning for the Detection of Oils Spills in Satellite Radar Images, Machine Learning, 1998, Vol. 30, pp. 195-219.

[25] Japkowicz, N., The Class Imbalance Problem: Significance and Strategies, Proceeding of the 2000 International Conference on Artificial Intelligence: Special Track on Inductive learning, 2000.

[26] Egan, J. P.,Signal detection theory and ROC analysis. Academic Press, London, 1975.

5.6 Appendix

5.6.1 R Codes

A typical example for the Abalone data set

```
#########################################################
#Nearest function using distance matrices
#########################################################
nearest <- function (X, n, k)
## Find k nearest neighbors of X[n, ] in the data frame
## or matrix X, utilizing function which.min k-times.
{
    N <- nrow(X)
    # inds contains the indices of nearest neighbors
    inds <- c(n); i <- 0
    while (i < k) {
        # use knn1 for one index...
    j =as.integer(which.min(d[-inds,n]))#d=1-aba.rf\$proximity
         #j <- as.integer(knn(X[-inds, ], X[n, ], 1:(N-length(
             inds))))
        # ...and change to true index of neighbor
        inds <- c(inds, setdiff(1:N, inds)[j])
        i <- i+1
    }
    # return nearest neighbor indices (without n, of course)
    return(inds[-1])
}

d<-as.matrix(dist(aba[,-8]))# for Euclidean Distances without
    Class
aba.rf <- randomForest(aba[,-8], ntree=500, mtry=3, proximity=
    TRUE) # for Random Forest Distances
d<-1-aba.rf\$proximity
# To find the number of nearest neighbor of minortiy class...
m<-2
k<-which(aba\$Rings=="G1")# indicate your minority class
NN<-c(3,5,7,9,11)
Complexity1<-0
aaa<-numeric(0)
for(m in 1:length(NN))
    {
aa<-numeric(0)
for (j in 0:NN[m]){
a<-numeric(0)
```

```
for (i in 1:length(k)) {
  at<-table(aba[nearest(aba[,-8],k[i],NN[m]),]\$Rings=="G1")["
      TRUE"]==j
  a<-c(a,at)
  }
  aa<-cbind(aa,a)
  }
  assign(paste("NN",NN[m],sep=""),c(length(which(is.na(aa[,1])))
      ,colSums(aa[,-1],na.rm=T)))
  assign(paste("Complexity",NN[m],sep=""),sum(get(paste("NN",NN[
      m],sep=""))[1:((NN[m]+1)/2)]))
  if (sum(get(paste("Complexity",NN[m],sep="")))-sum(get(paste("
      Complexity",NN[m]-2,sep="")))<(0.025*nrow(abamin))) stop(
      print(NN[m]-2))

  aaf<-aa
  }
#############################################################
#Replacing existing factor by complexity measure
#############################################################

k0<-which(is.na(aaf[,1]))
for (i in 1:(ncol(aaf)-1)){
assign(paste("k",i,sep=""),which(aaf[,i+1]==1))}

for (j in 0:(ncol(aaf)-1)){
wdbcnear<-numeric(0)
for (i in 1:length(get(paste("k",j,sep="")))) {
  wdbcnear<-rbind(wdbcnear,wdbc[nearest(wdbc[,-1],get(paste("k",
      j,sep=""))[i],NN[m]-2),])
  assign(paste("wdbc",j,sep=""),wdbcnear)
  }}
#############################################################
#R codes for SMOTE on Complexity
#############################################################
xsm<-numeric(0)
for (j in 0:(ncol(aaf)/2-1)){
nrpt=NN[m]-(2+j)
z0 <- d[get(paste("k",j,sep="")),get(paste("k",j,sep=""))]
```

```
set.seed(54)
#for (J in 1:nrpt) {
x<-abamin[get(paste("k",j,sep=""))
 for (k in 1:ncol(z0)) {
  xodr <- x[order(z0[,k],decreasing=F),]
  for (i in 1:nrpt) {
    xo <- ((xodr[1,]-xodr[i+1,])*runif(1))+xodr[1,] #copy
        without Class
    xo = cbind(xo,"G1")
    names(xo)[8] = "Rings"
    xsm <- rbind(xsm,xo) }}}

nrow(xsm)
#############################################################
#Over sampling using Complexity measurement
#############################################################
xcrnd<-numeric(0)
for (j in 0:(ncol(aaf)/2-1)){
nrpt=NN[m]-(2+j)
x<-abamin[get(paste("k",j,sep=""))]
set.seed(54)
#for (k in 1:nrpt) {
  nn = nrpt*nrow(x)
  samp <- sample(nrow(x),nn,replace=T)
  xo <-x[samp,]
  xcrnd = rbind(xcrnd,xo)  }
nrow(xcrnd)
```

Chapter 6

Feature Selection for Classification Problems with Imbalanced Data

Two challenges often faced in data mining are the presence of excessive or irrelevant features in a data set and unequal numbers of examples in the two classes in a binary classification problem. Most existing feature extraction methods are performed using a criterion function based on the classes. Although these methods work relatively well for balanced data, generally they are not optimal in any sense for imbalanced data sets. In this chapter, we propose a novel approach to feature selection for imbalanced data. This technique consists of an independent evaluation criterion based on a data complexity measure associated with the minority class. We first investigate the complexity measurement for the minority class in the feature space and find that there exist much better feature sets for which complexity for the minority class is relatively low. Then we propose an algorithm to find such features. Experiments show that the proposed algorithm consistently provides better results compared with the existing feature extraction algorithms, especially when the data set is highly imbalanced.

6.1 Introduction

Researchers have developed many techniques to try to overcome the Class Imbalance problem, including re-sampling, new algorithms and feature selection. Re-sampling and development of new algorithms are at the core of study to rectify the class imbalance problem, whereas feature selection is a relatively new development to combat class imbalance [1, 2]. In this chapter we will review the existing techniques of feature selection and their application in the Imbalance scenario. Research [1, 2, 3, 4] shows that in high dimensional data sets, feature selection can by itself combat the class imbalance problem. Our aim is to discover which feature selection technique will be the most effective in increasing the accuracy of the minority class. To this end we will apply existing techniques on the UCI datasets [5] to test their effectiveness. The UCI data sets are often used in the Imbalance data literature, with many researchers using these data sets to show the strength of their proposed technique. We avoid here the extreme feature selection problems, such as text classification [1] and microarray analysis [6] which have more features than cases. Our focus is on the performance of the techniques in selecting from a relatively modest set of possible feature when the main concern is the classification accuracy of the minority class.

In this chapter, we propose a novel feature selection algorithm using Complexity Measurement (CM) to evaluate the extracted low-level features. Using the best feature subset captured by Feature Selection using Complexity Measurement (FSCM), we compare the performance of the proposed framework with the performance of several other existing feature selection algorithms discussed above using benchmark data from UCI [5] under the well-known Classification Tree (CT) classifier. Overall, our proposed framework performs better than other feature selection methods over the classifier, and performs significantly better with highly imbalanced data sets.

The rest of the Chapter is organized as follows: Sections 6.2 and 6.3 provide a brief overview of the existing feature selection approaches and algorithms, Section 6.4 explains the motivation and methodology of our proposed algorithm, and Section 6.5 follows with the results of our experiments.

Finally, Section 6.6 gives conclusions and future work.

6.2 Current Feature Selection (FS) Approaches

Feature selection has been an ongoing research area in the classification, statistics, and data mining communities. The main aim of feature selection is to select a subset of input variables by eliminating features with little or no predictive information. Feature selection can significantly improve the efficiency of the resulting classifier models and often build a model that generalizes better to unseen points (test set). The advantages of feature selection are manifold, the most important being:

- to avoid overfitting and improve predictions;
- to gain deeper understanding of underlying processes that generate the data;
- to provide fast and comprehensive models.

It is often the case that finding the correct subset of predictive features is an important problem in its own right. For example, a physician may make a decision based on the selected medical tests/features whether a dangerous surgery is necessary for treatment or not. The search for a subset of relevant features introduces an additional complexity in the modeling task. Instead of just optimizing the parameters of the classifier for the full model, we now need to find the optimal features and then optimize the parameters for the optimal sub-model. There is no guarantee that optimal parameters for the full set will be equally optimal for the subsets [7]. Feature selection techniques are different from each other, in the way they incorporate the subsets into the model selections.

In the classification situation, feature selection can be divided into three categories: filter methods, wrappers and embedded methods. Table 6.1 provides a summary of feature selection techniques used in classification, the ad-

Table 6.1: Feature Selection Techniques

Methods	Advantages	Disadvantages	Methodology
Filter	Fast Scalable Independent of classifier	Ignores feature dependencies Ignore interaction with classifiers	χ^2 Information gain, gain ratio, symmetrical uncertainty Pearson Correlation Co-efficient(PCC) Relief (Algorithm)
Wrapper	Deterministic Simple Interacts with the classifier Models features dependencies Less computationally intensive than randomized methods	Risk of over fitting More prone than randomized algorithm to getting stuck in a local optimum Classifier dependent selection	Sequential forward selection Sequential backward elimination Branch and Bound algorithms
	Randomized Less prone to local optima Interacts with the classifier Models features Dependencies	Computationally intensive Classifier Dependent Selection Higher risk of over fitting than deterministic algorithms	Simulated annealing Randomized Hill Climbing Genetic Algorithm
Embedded	Interacts with the classifiers Better computational than wrapper methods	Classifier dependent selection	Random Forest Feature selection using the weight vector of SVM
PCA	Dimension reduction compress the data filter some of the noise in the data	not a feature selection method	Principal component analysis
Proposed Technique	Fast Independent of classifier simple Models feature dependencies works for class imbalance	computationally Intensive Risk of over fitting	Feature selection by Complexity measurement

vantages and disadvantages of each technique, and the most common methodologies.

A filter technique is univariate and assesses each variable individually on the specified criterion. The most common criterion is correlation with the classes where only those variables which have high correlation with the classes are selected for model building. Advantages of filters are that they can handle high dimensional data sets, they are computationally fast and simple and they are independent of the classifier algorithm. Therefore, they need to be performed only once, after which different classifiers can be applied. This feature selection technique is very popular in genetics and bioinformatics [8] and text classification [3]. The main disadvantage of this method is that it

ignores feature dependencies, which may lead to worse classification performance in comparison to the other feature selection techniques. An analysis of the CoIL Challenge 2000 by Elkan[9] found that feature selection metrics were not good enough for this task, instead the interaction between the different features also needed to be considered in the selection phase. The biggest mistake that he observed was that most feature selection methods did not select highly correlated variables, because they were considered to be redundant. Guyon and Elisseeff [10] show that apparently irrelevant features can be useful in conjunction with other features, and the combination of two highly correlated features can be even better than each feature independently.

Wrappers and embedded methods are feature selection method that consider interaction between the features during the selection process. In this setup, a search procedure in the space of possible subsets of features is defined, and various candidate subsets are generated and evaluated. The evaluation procedure for a specific subset is obtained by training and testing a specific classification algorithm, hence this technique is tailored to a specific classification model. To search all possible subsets of features a search algorithm is then wrapped around the classification model. The advantage of this technique is the interaction between features and classification algorithm. However, it suffers from two primary problems in high dimensional data sets. Firstly, the space for the feature subsets grows exponentially with the number of features; secondly, a subset selection method may find a feature selection that overfits the training data and produces even worse prediction than the baseline model [11]. In comparison to the Filter techniques, these techniques have a higher risk of overfitting and are computationally intensive, especially if the classifier has high computational cost.

In another class of the feature selection techniques, termed as embedded techniques, the search of the optimal subset of features is built into the classifier algorithm, such as Random Forest (see section 6.3.2.2). Just like the wrapper technique, this technique is tailored to a specific classification algorithm. The advantages of this technique is that it is less computationally intensive than the wrapper techniques.

6.2.1 Feature Selection in Class Imbalance

The curse of dimensionality tells us that if many of the features are noisy the cost of the classifier can be very high, and the performance may be extremely degraded [12]. If Class Imbalance is accompanied by high dimensionality of the data set [12], applying feature selection is clearly necessary. However we think that in the case of highly imbalanced data, feature selection should not be constrained to just high dimensional data sets, such as the cases of micoarray data and text classification, but used even for comparatively low dimensional data sets, to select better predictive features for the minority class. Forman [1] noted a similar observation for highly imbalance data sets and stated "no degree of clever induction can make up for a lack of predictive signal in the input space". In cases of class imbalance, when we are primarily interested in the accuracy of the minority class, the input space can be divided into class wise spaces, to reflect our bias in favor of the minority class. Researchers [1, 2, 3, 4] show that in high dimensional data sets, feature selection can alone combat the class imbalance problem. Our aim then is to use feature selection techniques to increase the classification accuracy of the minority class.

Feature selection has long been a part of machine learning and has been thoroughly researched [10], but its application in class imbalance problem is a relative recent development. Most of the research has been done in the last eight years and mostly for classifying text data sets using the filter methods mentioned in Table 6.1. Mladenic and Grobelnik [3] used the Yahoo web hierarchy and used filter feature selection methods. Forman [1] used the text sets from the TREC competitions, MEDLINE, OHSUMED and Reuters databases. Using the filter feature selection methodologies, he trained linear Support Vector Machine (SVM) and evaluated their performance using accuracy, precision, recall and F1-Measure. Zheng et al. [2] analyzed how the types of selected features affected classification performance. They used chi-square (CHI), information gain (IG) and odds ratio (OR) algorithms.

6.3 Feature selection Algorithms

In this section, we introduce those algorithms that we use later in our performance comparison. There are many feature selection algorithms available in the literature. Several of the most popular algorithms representing the various categories mentioned in Table 6.1 are selected for study here.

6.3.1 Filter-Based Feature Ranking Techniques

The procedure of feature ranking is to score each feature according to a particular criterion, based on which the best set of features is selected. The five standard filter-based feature ranking techniques used in this work include Statistics-based Feature Selection i.e., chi-square (CS), information gain (IG), gain ratio (GR) and symmetrical uncertainty (SU), as well as Relief algorithm introduced by Kira and Rendell [13]. The Statistics-based Feature Selection can handle either binary or nominal data. The UCI data sets [5] we study include continuous data, therefore we must pre-process the data before applying these techniques. We follow the method proposed by Guyon and Elisseeff [10]. First we find the mean feature value for the two classes, then we set a threshold at the midpoint between the two mean values. The feature is then binarized according to this threshold [10].

6.3.1.1 Chi-square

The Chi-square χ^2 (CS) test [14] evaluates features by ranking the chi-square statistic of each feature with respect to the class. The null hypothesis is that there is no correlation; i.e., each value is as likely to have examples in any one class as any other class. The χ^2 statistic,

$$\chi^2 = \sum_{i=1}^{n} \sum_{j=1}^{C} \frac{(O_{i,j} - E_{i,j})}{E_{i,j}} \tag{6.1}$$

measures how far away the actual value is from the expected value, assuming the null hypothesis. Here, n is the number of different values of the feature, C is the number of classes (in our work, C = 2), $O_{i,j}$ is the observed number

of examples with value i which are in class j, and $E_{i,j}$ is the expected number of examples with value i and class j. The larger this χ^2 statistic, the more likely it is that the distribution of values and classes are dependent; that is, that the feature is relevant to the class.

6.3.1.2 Entropy based techniques

An Information Entropy Function is a measure of the uncertainty associated with a random variable. Information gain, gain ratio, and symmetrical uncertainty are measures based on the concept of entropy from Information Theory [15]. The Information gain (IG) measure scores features by computing their information gain with respect to the class. IG is the information provided about the target class attribute Y, given the value of another attribute X. IG measures the decrease of the weighted average impurity of the partitions compared to the impurity (a measure of diversity for the outcome in a specific set of nodes) of the complete set of data. A drawback of IG is that it tends to prefer attributes with a larger number of possible values, i.e., if one attribute has a larger number of distinct values, it will appear to gain more information than those with fewer values, even if it is actually no more informative. One strategy to solve this problem is to use the gain ratio (GR), which penalizes multiple-valued attributes. Symmetrical uncertainty (SU) is another way to overcome the problem of IGs bias toward attributes with more values, by dividing by the sum of the entropies of X and Y.

6.3.1.3 Relief

Relief is a case-based feature ranking technique introduced by Kira and Rendell [13]. Relief evaluates each feature by its ability to distinguish the neighboring cases. It randomly samples the examples and checks the cases of the same and different classes that are near to each other. An exponential function governs how rapidly the weights degrade with the distance. ReliefF [16] is an extension of the Relief algorithm that can handle noise and multiclass data sets, and is implemented in the R package FSelector [17].

6.3.2 Wrapper-based Feature Selection Techniques

Feature selection using a wrapper provides an alternative way of selecting feature subsets, incorporating the classifier into the search and thus selecting more efficient subsets. Wrapper technique uses the method of classification to measure the importance of subsets of features; hence the features selected depend on the classifier technique used. Wrapper methods generally result in better performance than filter methods because the feature selection process provides opportunities to construct a more accurate classifier. However, wrapper methods are highly computational for large dimensional databases since all possible feature subsets must be evaluated with the classifier algorithm used. We test two very popular wrapper-based techniques in this work, Genetic Algorithm (GA) [18] and Random Forest [19].

6.3.2.1 Genetic Algorithm (GA)

Genetic Algorithm (GA) [20] is a random search method that can effectively explore large search spaces, which is necessary in the case of feature selection. Further, unlike many search algorithms, which perform a local, greedy search, GA performs a global search. A genetic algorithm (GA) is a search algorithm inspired by the principle of natural evolution. The basic idea is to evolve a population of individuals, where each individual is a candidate solution to a given problem. GA mainly comprises of three operators: reproduction, crossover, and mutation. Reproduction selects good strings; crossover combines good strings to try to generate better offspring; mutation alters a string locally to attempt to create a better string. In each generation, the population is evaluated and tested for termination of the algorithm. If the termination criterion is not satisfied, the population is operated upon by the three GA operators and then re-evaluated. This procedure is continued until the termination criterion is met. In our application of GA, the population evaluation criterion is to minimize the error rate for the minority class.

6.3.2.2 Random Forest (RF)

A Random Forest [21] is an ensemble method that combines several individual classification trees in the following way: several bootstrap samples are drawn from the training set, and an unpruned classification tree is constructed for each bootstrap sample. During construction of each tree, at each node, a small random subset of predictor variables or features are tried to split that node. From the complete forest the status of the response variable is predicted as an average or majority vote of the predictions of all trees. Random forests can greatly increase the prediction accuracy as compared to individual classification trees. Unpruned trees have low bias and high variance but the variance is reduced by averaging the bootstrapped trees. The interpretation of a random forest is not as straightforward as that of an individual classification tree, where the influence of a predictor variable directly corresponds to its position in the tree. Thus, alternative measures for variable importance are required for the interpretation of random forests.
A simple variable importance measure to use in tree-based ensemble methods is to just count the number of times each variable is selected by all individual trees in the ensemble. More complex variable importance measures incorporate a (weighted) mean of each individual tree's improvement in the splitting criterion produced by each variable [22]. An example in classification for such a measure is the "Gini index" available in random forest implementations. The Gini index describes the improvement in the "Gini gain" splitting criterion. Another variable importance measure available in random forests is the "permutation accuracy importance" measure. It is based on the following idea: by randomly permuting the predictor variable X_j, it should no longer have an association with the response Y. When the permuted variable X_j, together with the remaining predictor variables (unpermuted), is used to predict the response, the prediction accuracy (i.e. the number of observations classified correctly) decreases significantly, if the original variable X_j was associated with the response. Thus, variable importance is measured by the difference in prediction accuracy before and after randomly permuting X_j. Random Forest permutation accuracy importance has advantages over

filter methods (for feature selection) in that it covers the impact of each predictor variable individually as well as in multivariate interactions with other predictor variables.

6.3.2.3 Boruta Algorithm

Here we discuss the main theme of the Boruta Algorithm; for more details see Kursa and Rudnicki [23]. Boruta Algorithm is a wrapper built around the random forest classification algorithm [19]. It uses Z score as an importance measure of a feature attribute, obtained as the loss of accuracy of classification caused by random permutation since it takes into account the fluctuations of mean accuracy loss among trees in the forest. Since Z scores can not be computed directly, feature importance returned by the Random Forest algorithm does not have a standard normal distribution [24]. The Boruta Algorithm has the following steps.

1. Extend the information system by adding copies of all variables (the information system is always extended by at least 5 shadow attributes, even if the number of attributes in the original set is lower than 5).

2. Shuffle the added attributes individually to remove their correlations with the response.

3. Run a random forest classifier on the extended information system and gather the Z scores computed.

4. Find the maximum Z score among shadow attributes (MZSA), and then assign a hit to every attribute that scored better than MZSA.

5. For each attribute with undetermined importance perform a two-sided test of equality with the MZSA.

6. Deem the attributes which have importance significantly lower than MZSA as 'unimportant' and permanently remove them from the information system.

7. Deem the attributes which have importance significantly higher than MZSA as 'important'.

8. Remove all shadow attributes.

9. Repeat the procedure until the importance is assigned for all the attributes, or the algorithm has reached the previously set limit of the Random Forest runs.

We use Boruta Algorithm instead of RF variable importance, due to fact that RF will give just a ranking of variables, after which we make a subjective decision on the number of variables to be used, whereas Boruta Algorithm will recommend variables deemed important . Moreover we want to analyze the efficiency of Boruta Algorithm in imbalanced data sets.

6.3.3 Transformation-based Feature Selection

Principal Components Analysis (PCA) transforms the set of features to the eigenvector space. Since each eigenvalue gives the variance along the corresponding axis, we could use such a special coordinate system that depends on the cloud of points with a certain variance in each direction. All the components could be used as new features but the first few would account for most of the variance in the data set. Rather than pick a subset of features, this method picks a small set of linear functions of the features. The basic idea when using PCA as a tool for feature selection is to select variables according to the magnitude (from largest to smallest in absolute values) of their coefficients (loadings). PCA seeks to replace p (more or less correlated) variables by $f < p$ uncorrelated linear combinations (projections) of the original variables. The f principal components are ranked by importance through their explained variance, and each variable contributes with varying degree to each component. Using the largest variance criteria would be akin to feature extraction, where principal components are used as new features, instead of the original variables. However, we can decide to keep only the first component and select the $m < p$ variables that have the highest absolute coefficient; the number m might be based on a proportion of the number of variables (e.g.,

keep only the top 10% of the p variables), or a fixed cutoff (e.g., considering a threshold on the normalized coefficients). We select a fixed cutoff of at least 90% of total variation accounted for by m principal components.

6.4 Proposed Algorithm

For classification, it may be optimal to use the minimum achievable error, the Bayes error, as a criterion [25]. However, the Bayes error cannot be easily expressed in an analytical form except in special cases. Instead, an estimate of the Bayes error is often used as a criterion. For instance, wrapper techniques are based on the minimum classification error, where the classification error is measured by misclassification over the training samples. The nearest neighbor technique is used in the literature [26] to estimate Bayes error. We have thoroughly studied the k-NN approach to estimate Bayes error in the class imbalance scenario, focusing on the local information for each data point in the minority class via nearest neighbors, and using this information to define a data complexity measure, known as Complexity Measurement (CM). We have found a strong correlation between CM and misclassification of the minority class. Based on this assumption, we propose to approximate the classification error using the CM as a minimization criterion. A benefit of this approach is that the number of features necessary for classification without serious information loss can be predicted (please see section 6.5.2 for more detail). The proposed method is designed to deal with feature selection for imbalanced data, focusing specifically on accuracy of the minority class. The proposed methodology consists of two basic steps.

1. We are interested in the accuracy of the minority class, hence we will measure the complexity of the minority class.

2. Only those features will be selected which minimize the overall complexity of minority class.

We begin the development of the feature extraction algorithm by considering two-class problems. It can be extended to multiclass problems using a 1-vs-rest approach. Let $X_{p\times 1}$ be an observation vector and ω_l a feature selection

vector, such that a single feature is $\omega_l^t X_{p\times 1}$, where sum of the elements Σ_{p^*} $\omega_{lp^*} = 1$ and $\omega_{lp^*} \in \{0, 1\}$ (except for PCA). Matrix $\omega = (\omega_1, \omega_2, \ldots, \omega_{\acute{p}})_{p\times\acute{p}}$, transforms an observation vector X in the p-dimensional space into a new $\acute{p}$-dimensional feature vector F,

$$F = \omega^t X = (\omega_1, \omega_2, \ldots, \omega_{\acute{p}})^t X \tag{6.2}$$

where $\acute{p} \leq p$. Consequently, the estimate of the classification error in the subspace depends on the matrix ω. In other words, the criterion is given by

$$J_\omega = CM_{j,k,\omega} \tag{6.3}$$

where $CM_{j,k,\omega}$) is an estimate using k-NN of the classification error based on the CM in the subspace spanned by ω for class j. Here we take j to be the minority class, and determine k as in section 3.5.0.1, so we abbreviate to CM_ω. In order to find a solution for ω that minimizes J_ω, we use an algorithm that employs heuristic search.

6.4.1 Heuristic Search

Searching the space of feature subsets within reasonable time constraints is essential, if a feature selection algorithm is to operate on data with a large number of features. One simple search strategy, called greedy hill climbing, considers local optimum to the current feature subset. Often, a local optima is simply the addition or deletion of a single feature from the subset. When the algorithm considers only additions to the feature subset it is known as forward selection; considering only deletions is known as backward elimination [27, 28].

6.4.1.1 Sequential Forward Selection Using CM

Starting from the empty set, sequentially add the feature x^+ that results in the lowest objective function $J(\omega_l + x^+)$ when combined with the features ω_l that have already been selected.

Algorithm: Sequential Forward Selection

1. Start with the empty set $\omega_l = \emptyset$, and $l = 0$
2. Select the next best feature; $x^+ = argmin_{x^+ \notin \omega_l}[J(\omega_l + x^+)]$.
3. Update $\omega_{l+1} = \omega_l + x^+; l = l + 1$.
4. Store $CM_{l+1} = J_{(\omega_{l+1})}$.
5. Go to 2.

The minimum CM_l will determine ω. We have found that the Forward selection method performs best when the optimal subset has a small number of features; The main disadvantage of this is that it is unable to remove features that become obsolete after the addition of other features.

6.4.1.2 Sequential Backward Selection Using CM

Sequential Backward Selection works in the opposite direction of sequential forward selection. Starting from the full set, sequentially remove the feature x^- that results in the smallest increase in the value of the objective function $J(\omega_l - x^-)$. Notice that removal of a feature may actually lead to a decrease in the objective function $J(\omega_l - x^-) < J(\omega_l)$.

Algorithm: Sequential Backward Selection

1. Start with the full set $\omega_0 = X$.
2. Remove the worst feature $x^- = argmin_{x^- \in \omega_l}[J(\omega_l - x^-)]$.
3. Update $\omega_{l+1} = \omega_l - x^-; l = l + 1$.
4. Store $CM_{l+1} = J_{(\omega_{l+1})}$.
5. Go to 2.

Subspace/subset feature ω will correspond to the minimum value of CM_i, i.e., those features will be selected which give the minimum of CM_l. Backward selection works best when the optimal feature subset has a large number of features, since backward selection spends most of its time visiting large subsets. The main limitation of backward selection is its inability to re-evaluate the usefulness of a feature after it has been discarded.

Another option for the sequential selection is stepwise forward and backward selection, i.e., if we remove a particular feature and our evaluation criterion increases, so we should not remove this particular feature. However stepwise forward and backward selection was not working in our situation. If we added a particular feature, our algorithm again selected this feature for removal. Hence feature selection was not going anywhere. Instead of getting local optima we are more interested in getting global optima, so we prefer sequential forward and backward selection so we can go through all the features and selected those features which are having favorable multivariate interactions with other features, by providing global minimum value of CM_l.

Despite respective disadvantages of the sequential backward or forward selection, we are content with the procedure due to following reasons.

1. The only way to ensure the best possible selection is to evaluate all possible combination of features, which is not computationally feasible if there are a large number of features.

2. The above procedure is quick for up to moderate number of features, e.g, 40. For higher dimension parallel computing could be used.

3. For variable selection purposes the advantage of the FSCM as compared to filter based methods is that it covers the impact of each predictor variable individually as well as in multivariate interaction with other predictor variables.

4. Its uses an independent evaluation criterion under the wrapper method, hence once the features have been selected, these can be used with any classifier.

For feature selection in the example consider below, we fix $k = 5$, to make the process quick while giving good results. But different values of k could be used according to the nature of the data set.

6.5 Experiments:

6.5.1 Artificial Data Set:

We have used many simulated data sets in order to test our algorithm, however we found one simulation study particularly interesting. In this study the four variables in the Iris data set are complemented with 36 random variables. Variable selection should find the four original variables back, an example used by Ron Wehrens [29]. When we applied our proposed methodology for various classes of Iris data sets, only one feature was found interesting to best discriminate the classes. We plotted a scatter plot for this variable, shown in Fig 6.1. Comparison is also made with the Multidimensional Scaling of the original data sets.

From Fig 6.1 left panel it is clear that Iris class 'setosa' is quite different from the other classes, and this is a reason that no classifier misclassified this class. For the classes 'versicolor' and 'viriginica' we can see some overlap along the y-axis, and this is a reason that we find misclassification from 'versicolor' to 'viriginica'. Note that the x-axis is arbitrary in the left panel of Fig 6.1. We can visualize the Iris data set by MDS on Euclidean distance from Fig 6.1 right panel, where we can see the same pattern along the x-axis, but no pattern along the y-axis. We measure the stress function to show how well the 2-dimension MDS plot represents the data, stress= 2.813, which is expected, as this is a low dimensional data set (i.e., 4 dimension only), so stress is small, which warrants the representation by MDS plot. Hence we can argue that only one dimension is necessary to visualize the classes and this dimension is 'Petal.width'. Moreover, there was a no loss of information, when we use a learning algorithm for classification using this variable only.

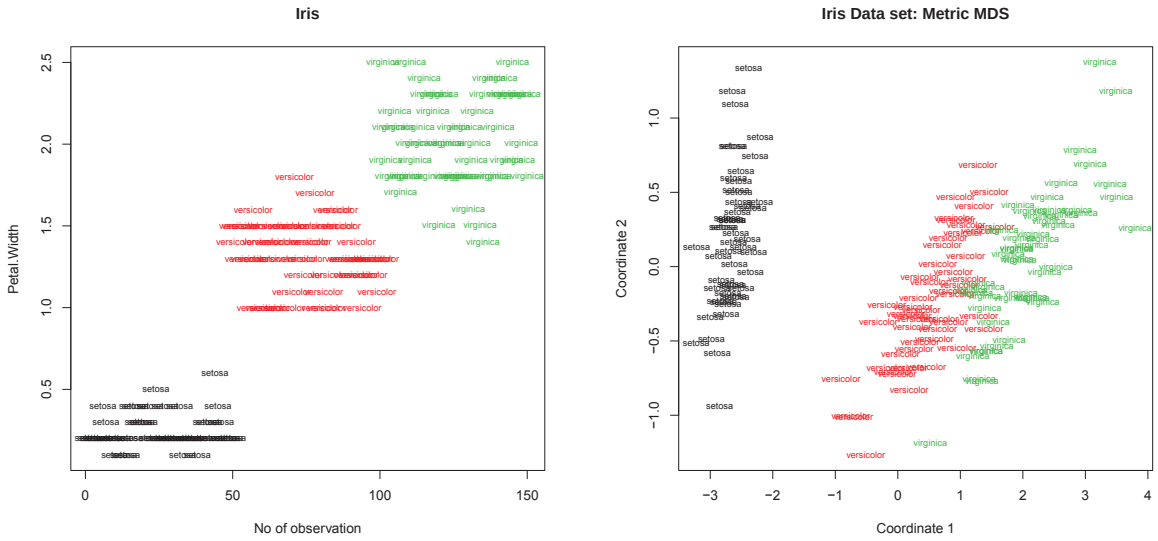

Figure 6.1: Visualization by One dimension selected by proposed algorithm (left) and MDS (right) for UCI Iris data set. The names and color combination shows different classes.

6.5.2 Real Data Sets

To compare the performance of existing feature selection methods and our proposed methodology with real data, we have used 9 data sets from the UCI data repository [5]. Information about these data sets is summarized in Table 6.2. All analyses were done using 5-fold cross validation with the resulting sensitivity values averaged to a single score. We use CM as the general indicator of whether a data set is easy or difficult to learn, and the examples are arranged accordingly.

All programs were written in the statistical environment R [30], and relevant R codes for our algorithm are provided in the Appendix 6.7. For the construction of tree models we used the `RPART` package [31], for nearest-neighbor based on Euclidean distance we used R package `class` [32] with a little modification of the input and output, and to compute Random Forest distances we used the R package `randomForest` [21]. For Feature Selection we used R package `FSelector` [17] for filter feature selection method, R package `genalg` [29] for Genetic Algorithm with fitness function set to improve

Table 6.2: Description of UCI data sets

Data Set	Feature(p)	Size	Target	Size of Min	Size of Maj	%Min	CM	Sensitivity
WDBC	30	569	malignant	212	357	37.3	0.080	0.88
Ionosphere	34	351	bad	126	225	35.9	0.405	0.62
Pima	8	768	class 1	268	500	34.9	0.427	0.62
Vehicle	16	846	opel	212	634	25.1	0.566	0.56
WPBC	31	198	recur	47	151	23.7	0.745	0.44
Yeast	8	1484	ME3	163	1321	11.0	0.325	0.72
Abalone2	8	4177	Ring<7	448	3729	10.7	0.489	0.59
Satimage	37	6435	class 4	626	5809	9.7	0.345	0.57
Abalone1	8	4177	Ring=7	391	3786	9.4	0.903	0.2

Data sets	Methodology	Feature	Accuracy	G Mean	Sensitivity	Specificity	AUC
WDBC	Nil	30	0.920	0.910	0.880	0.940	0.910
	FSCM	6	0.948	0.939	0.914	0.966	0.940
	RF(B)	30					
	GA	5	0.946	0.938	0.914	0.963	0.939
	Chi	6	0.926	0.915	0.876	0.955	0.916
	IG	6	0.926	0.915	0.876	0.955	0.916
	GR	6	0.924	0.912	0.871	0.955	0.913
	SU	6	0.924	0.913	0.876	0.952	0.914
	RReliefF	6	0.944	0.938	0.919	0.958	0.938
	PCA	7PC	0.948	0.941	0.914	0.969	**0.942**

Table 6.3: Sensitivity and AUC comparison of Feature Selection methods applied to WDBC data sets. Rows gives the various Feature Selection methodologies is used. Figures gives mean values calculated using stratified 5 fold cross validation

sensitivity, and R package `Boruta` [23] for Boruta Algorithm.

6.5.3 Results:

We consider the data sets ranked according to the CM measurement. The Winconsin Diagnostic Breast Cancer (WDBC) from the UCI Machine Learning Repository has 32 variables computed from a digitized image describing the characteristics of the cell nuclei present in each of 569 images. The class variable of interest here is the diagnosis, either benign (majority class) or malignant (minority), with particular emphasis on the detection of the minority class. Excluding patient ID and diagnosis (class label) columns, we considered 30 features for our analysis.

Data sets	Methodology	Feature	Accuracy	G Mean	Sensitivity	Specificity	AUC
Yeast	Nil	8	0.940	0.830	0.720	0.970	**0.845**
	FSCM	2	0.942	0.834	0.719	0.970	**0.845**
	RF(B)	5	0.942	0.827	0.706	0.970	0.838
	GA	3	0.880	0.000	0.000	0.985	0.492
	Chi	1	0.922	0.808	0.688	0.952	0.820
	IG	3	0.942	0.806	0.669	0.973	0.821
	GR	3	0.942	0.806	0.669	0.973	0.821
	SU	3	0.942	0.806	0.669	0.973	0.821
	RReliefF	6	0.942	0.827	0.706	0.969	0.838
	PCA	6	0.922	0.770	0.625	0.955	0.790

Table 6.4: Sensitivity and AUC comparison of Feature selection methods applied to Yeast data sets. Rows gives the various Feature selection methodologies is used. Figures gives mean values calculated using stratified 5 fold cross validation

Table 6.3 illustrates the results using the sensitivity (accuracy of minority class) and AUC. Feature stands for number of feature used in the model. Note that in all of the nine methodologies used, only Boruta algorithm failed to select important features, as it considered all features as equally important. Whereas our proposed algorithm (FSCM), Genetic Algorithm (GA) and PCA achieved better results than the original Classification Tree, with PCA the winner based on AUC value (the best results are marked in bold). Overall we can say that feature selection was successful in this data set by increasing the accuracy of minority class by 3%.

Consider next the Yeast data set, which concerns determination of protein cellular localization sites. This data set has 1484 observations with 8 feature and 7 classes. We select the class of sequence types ME3 (membrane protein, no N-terminal signal with 163 examples) vs Rest reducing the classification problem to a binary decision. This is a small data set with regard to standard feature selection, but has the advantages that we can easily evaluate all possible subsets.

Table 6.4 gives the results obtained from the various feature selection methodologies. All methodologies were able to identify the important feature except GA which failed in this data set (no detection for minority class). The variables proposed by the GA were not able to predict any of the minority class. This also reflects the restive nature of the randomized feature selection

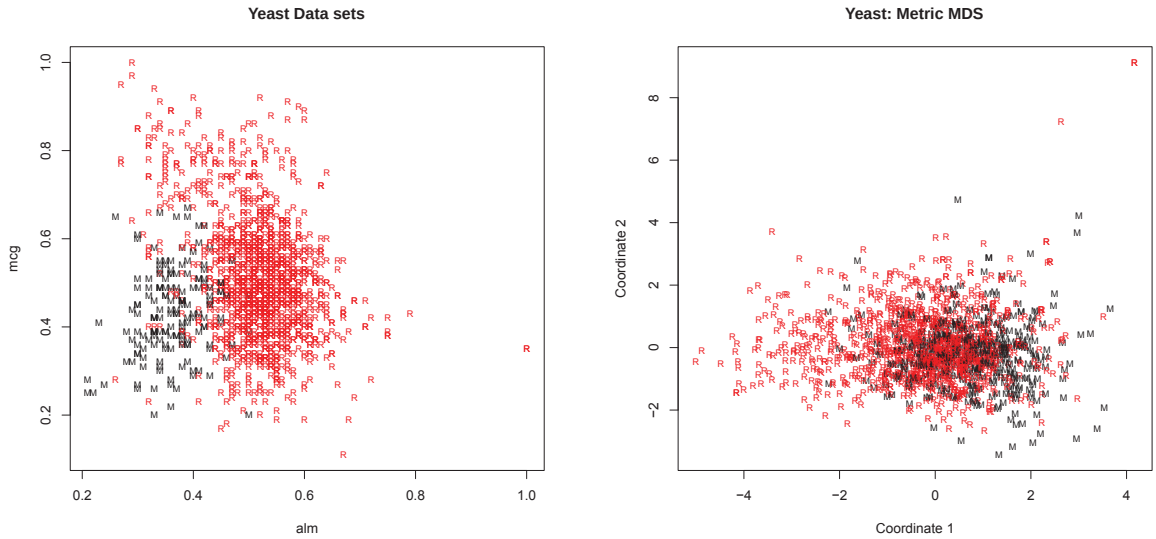

Figure 6.2: Visualization by 2 dimension selected by proposed algorithm (left) and MDS (right) for UCI yeast dataset. The symbols and color combination shows two different classes: M (ME3) minority class and R (Rest of classes: majority class).

techniques like GA. We need a lot of iterations to get any sensible model. Of particular interest is the feature selection by our proposed algorithm, in which only two out of the 8 features were selected without loss of information. If we plot these two features (Figure 6.2) we can see clear separation between the two classes (left panel of Fig 6.2), along the two axes; x-axis (alm: Score of the ALOM membrane spanning region prediction program) and y-axis; (mcg: McGeoch's method for signal sequence recognition). We can see some overlap in the datasets, due to which there are a number of false negative cases, which results in a fall in accuracy of minority class. If we compare with the Multidimensional Scaling (MDS) visualization, we can see a cluster of minority class (black), but no clear separation between the classes. The stress function for this MDS presentation is 36.243, which is very high, and due to this reason we can not see any clear separation between the classes.

Turning now to some highly imbalanced data sets, Satimage and Abalone (minority class $< 10\%$), once again our proposed algorithm FSCM proves to be a consistent technique performing really well in these cases as well. Table

6.5 gives the results.

Data sets	Methodology	Feature	Accuracy	G Mean	Sensitivity	Specificity	AUC
Sat	Nil	37	0.920	0.741	0.573	0.957	0.765
	CM	28	0.920	0.763	0.611	0.952	**0.782**
	RF(B)	37					
	GA	6	0.92	0.653	0.436	0.978	0.703
	Chi	18	0.900	0.716	0.545	0.941	0.743
	IG	18	0.890	0.702	0.526	0.938	0.732
	GR	18	0.900	0.727	0.564	0.937	0.750
	SU	18	0.900	0.743	0.588	0.940	0.764
	RReliefF	18	0.920	0.748	0.583	0.959	0.771
	PCA	8	0.880	0.730	0.578	0.921	0.749
Abalone	Nil	8	0.880	0.430	0.200	0.940	0.570
	CM	5	0.874	0.455	0.221	0.942	**0.581**
	RF(B)	8					
	GA	3	0.878	0.396	0.167	0.952	0.560
	Chi	8					
	IG	8					
	GR	8					
	SU	8					
	RReliefF	7	0.870	0.435	0.205	0.938	0.571
	PCA	2	0.880	0.443	0.208	0.947	0.577

Table 6.5: Sensitivity and AUC comparison of Feature selection methods applied to highly imbalanced Satimage and Abalone data sets. Rows gives the various Feature selection methodologies is used. Figures gives mean values calculated using stratified 5 fold cross validation

For the Satimage data set our algorithm selected 28 out of the 37 possible features, which increased the accuracy of the minority class by 4%. Wrapper based method Boruta and GA were not able to provide any significant features, maybe due to randomization and these algorithms needing high numbers of iterations to provide any sensible solutions. Whereas filter selection methods were able to half the number of features, without much decrease in the AUC values. PCA proved to be useful, since only accuracy of the majority class suffered. For the Abalone data set, our proposed algorithm performed well, by increasing the accuracy of the minority class by 2%, whereas most of the other feature selection methods failed to deliver. PCA once again proved to be useful. Detailed results for all data sets described in the UCI archive Table 6.2 are given in the Appendix.

Our proposed algorithm Feature Selection using Complexity Measure-

ment (FSCM) performs consistently well in all UCI data set (moderate to highly imbalanced). In terms of AUC values, FSCM often works better than other feature selection techniques as well as the model using all the features. In terms of sensitivity values sometimes it shows dramatic improvement. This is probably due to fact that FSCM is focusing primarily on reducing the noise for the minority class. Moreover, using CM as the criterion, we develop an independent criterion of feature selection for wrapper selection, which is lacking in the literature [8]. Hence the features once selected can be used with different classifiers to build a number of classification models.

6.6 Discussion and Future Work:

Feature selection for highly imbalanced data is a major challenge in Data Mining and Statistics. We have proposed a sequential feature selection method using complexity measure (CM) to deal with this problem. To evaluate the effectiveness of the proposed method, we have applied our technique on the well known UCI data sets from the class imbalance literature. We use the complexity measurement (CM) criterion which is independent of the classifier hence selected features can be used with different classifiers to build a number of classification models. Using Feature Selection using Complexity Measurement (FSCM) as a criterion we can specifically focus on the minority class, hence those features (and multivariate interactions between predictors) can be selected, which form a better model for the minority class. We compared the new method to three approaches - (i) Filter based feature ranking (ii) Wrapper based feature feature selection and (iii) Dimension reduction. The experimental results demonstrate that (1) our proposed method performed consistently better than the other approaches; and (2) the performance strength of our proposed method over other methods becomes increasingly evident as the class imbalance becomes more serious. The conclusions obtained in this chapter regarding the effectiveness of the proposed feature selection approach are based on the experiments conducted on data sets from UCI data sets. Future work will involve more experiments on data sets from different domains. Moreover, in order to improve the feature selec-

tion of existing techniques we will consider data sampling when the data sets have unequal numbers of examples in the two classes; different data sampling techniques can also be considered.

References

[1] Forman, G.and Cohen, I., Learning from Litte: Comparison of Classifiers Given Little Training, Proceeding 8th European conference, principles and practice of knowledge Discovery in Database, pp.161-172, 2004.

[2] Zheng, Z., Wu, X., and Srihari, R., Feature Selection for Text Categorization on Imbalanced Data, ACM SIGKDD Explorations Newletter, Vol. 6, pp. 80-89, 1999.

[3] Mladenic, D., and Grobelnik, M., Feature Selection for Unbalanced Class distribution and Naive Bayes, Proceeding of 16th international conference on Machine learning, pp. 258-267, 1999.

[4] Chen, X. and Wasikowski, M., Fast: A ROC-based Feature Selection metric for small samples and Imbalanced data classification Problems, Proceeding of ACM SIGKDD '08, pp. 124-133, 2008.

[5] UCI KDD archive.http://kdd.ics.uci.edu/databases/covertype/covertype.html. 2005.

[6] Xiong, H. and Chen, X., Kernel Based Distance Metric Learning for Microarray Data Classification, BMC Bioinformatics, Vol. 7, no. 299, pp. 1-11, 2006.

[7] Daelmans, W., Hoste, V., Meulder, F. D., and Naudts, B., Combined optimization of feature selection and algorithm parameter interaction in machine learning of language, In proceeding of the 14th European Conference on Machine learning, pp. 84-95, 2003.

[8] Somorjai, R., Dolenko, B., and Baumgartner, R. Class prediction and discovery using gene microarray and proteomics mass spectroscopy data: curses, caveats, cautions. Bioinformatics, Vol. 21, pp. 631-643, 2003.

[9] Elkan, C., Magical Thinking in Data Mining: Lesson from CoIL Challenge 2000, Proceeding ACM SIGKDD'01, pp. 426-431, 2001.

[10] Guyon, I. and Elisseeff, A., An Introduction to Variable and Feature Selection, Journal of Machine learning Research, Vol.3, pp. 1157-1182, 2003.

[11] Loughrey, L., and Cunnuingham, P., Overfitting in Wrapper-based Feature Selection, Journal of machine learning reserach, Vol. 3, pp.1157-1182, 2003.

[12] Chawla, N., Japkowicz, N., and Kotcz, A., Editorial: Specail Issue on learning from Imbalanced data sets, ACM SIGKDD Explorations Newsletter, Vol. 6, no. 1, pp. 1-6, 2004.

[13] Kira, K., and Rendell, L. A., A approach to feature selection, in proceeding of 9th international workshop on machine learning, pp. 249-256, 1992.

[14] Plackett, R. L., Karl Pearson and the chi-squared test, International Statistical Review, Vol.51, no. 1, p.59-72, 1983.

[15] Witten, I. H., and Frank. E., Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Mateo, California, 2nd edition, 2005.

[16] Prados, J., et al., Mining mass spectra for diagnosis and biomaker discovery of cerebral accidents, Proteomics, Vol. 4, pp. 2320-2332, 2004.

[17] Romanski, P., FSelector (http://cran.r-project.org/web/packages/FSelector/index.html).

[18] Goldberg, D., Genetic Algorithms in Search, Optimization, and Machine learning, Addison Wesley, 1989.

[19] Breiman, L., Random Forest, Machine Learning, VOl. 45, no. 1, pp. 5-32, 2001.

[20] Holland, J. , Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, Michigan, 1975.

[21] Breiman, L., Random Forest, Machine Learning, Vol. 45, no. 1, pp. 5-32, 2001.

[22] Friedman, J: Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics, Vol. 29, pp. 1189-1232, 2001.

[23] Kursa, M. B. and Rudnicki, W.R., Feature Selection with the Boruta Package. Journal of Statistical Software, Vol. 36, no.11, pp. 1-13, 2010. URL http://www.jstatsoft.org/v36/i11/.

[24] Rudnicki WR, Kierczak M, Koronacki J, Komorowski J, A Statistical Method for Determining Importance of Variables in an Information System." In S Greco, H Y, S Hirano, M Inuiguchi, S Miyamoto, HS Nguyen, R Slowinski (eds.), Rough Sets and Current Trends in Computing, 5th International Conference, RSCTC 2006, Kobe, Japan, November 6-8, 2006, Proceedings, volume 4259 of Lecture Notes in Computer Science, pp. 557-566. Springer-Verlag, New York, 2006.

[25] Fukunga, K., Introduction to Statistical Pattern Recognition, 2nd edition. Boston: Academic Press, 1990.

[26] Cover, T. M., and Hart P. E., Nearest Neighbor pattern classification. IEEE Transactions on Information Theory, Vol. 13, pp. 21-27, 1967.

[27] Kittler, J. Feature set search algorithms. In C. H. Chen, editor, Pattern Recognition and Signal Processing. Sijhoff an Noordhoff, the Netherlands, 1978.

[28] Miller, A. J. Subset Selection in Regression. Chapman and Hall, New York, 1990.

[29] Willighagen, E., Genetic Algorithm, http://cran.r-project.org/web/packages/genalg.

[30] R Development Core Team, R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, http://www.r-project.org, 2004.

[31] Therneau, T.M., Atkinson B., and R port by Brian Ripley, rpart: Recursive Partitioning. R package version 3.1-42, (http://www.mayo.edu/biostatistics), 2004.

[32] Ripley, B., class: Function for Classification. R package version 7.3-42, http://www.stat.ox.ac.uk/pub/MASS4/, 2010.

6.7 Appendix

6.7.1 R Codes

```
############################################################
# Nearest function for the Complexity
############################################################
nearest <- function (X, n, k=5)
## Find k nearest neighbors of X[n, ] in the data frame
## or matrix X, utilizing function which.min k-times.
{
    N <- nrow(X)
    # inds contains the indices of nearest neighbors
    inds <- c(n); i <- 0
    while (i < k) {
        # use knn1 for one index...
    j =as.integer(which.min(d[-inds,n]))#d=1-wdbc.rf$proximity
         #j <- as.integer(knn(X[-inds, ], X[n, ], 1:(N-length(
             inds))))
        # ...and change to true index of neighbor
```

```
        inds <- c(inds, setdiff(1:N, inds)[j])
        i <- i+1
    }
    # return nearest neighbor indices (without n, of course)
    return(inds[-1])
}


#####################################################
# main loop for variable elimination back ward #
#####################################################
#Let
x<-X
rvar<-numeric(0)
rec<-numeric(0)
for (N in 1:ncol(X)){

  #get complexity measurement/ data errors
d<-as.matrix(dist(x))
a<-numeric(0)
k<-which(Y=="versicolor")#define your minority class
for (i in 1:length(k)) {
   at<-table(Y[nearest(x,k[i])])
   a<-rbind(a,at)
   }
a<-as.matrix(a)
overallVariableImportance=length(which(a[,2]<3))
com<-numeric(0)
for(I in 1:ncol(x)){
d<-as.matrix(dist(x[,-I]))
a<-numeric(0)
k<-which(Y=="versicolor")
for (i in 1:length(k)) {
   at<-table(Y[nearest(x,k[i])])
   a<-rbind(a,at)
   }
a<-as.matrix(a)
com<-c(com,length(which(a[,2]<3)))}
VariableImportance<-com
```

```
   #remove the worst variable
  Z <- order(VariableImportance,decreasing = FALSE)
  IND <-Z[1] #seems the target will always be index 1
  var_to_remove <- names(x[IND])
  x[IND] = NULL
rec<-c(rec,min(VariableImportance))
rvar<-c(rvar,var_to_remove )
  #report
  cat("\nOverallComplexity ",overallVariableImportance)
  cat("\nComplexity ",min(VariableImportance))
  cat("\nremoving variable ", var_to_remove)
  flush.console()
}
names(x)
ncol(x)
rvar
rec
cat("\nSelected variables:", rvar[-c(1:which(rec==min(rec))[
    length(which(rec==min(rec)))])])
#####################################################
# main loop for variable elimination forward #
#####################################################
#Let
X<-data.frame(X)
xf<-X
x<-NULL
rvar<-numeric(0)
rec<-numeric(0)
for (N in 1:ncol(X)){
 #overallVariableImportance=complexity(y.1,x)
com<-numeric(0)
for(I in 1:ncol(xf)){
d<-as.matrix(dist(cbind(x,xf[,I])))#[,c(vs,I+C)]))
a<-numeric(0)
k<-which(Y=="versicolor")#define your minority
for (i in 1:length(k)) {
  at<-table(Y[nearest(cbind(x,xf[,I]),k[i])])
  a<-rbind(a,at)
  }
```

```
a<-as.matrix(a)
com<-c(com,length(which(a[,2]<3)))}
VariableImportance<-com
      yt<-x
  #add the best variable
  Z <- order(VariableImportance,decreasing = FALSE)
  IND <-Z[1] #seems the target will always be index 1
  var_to_add <- names(xf[IND])
rvar<-c(rvar,var_to_add)
  x = X[,rvar]
  xf=xf[,-IND]
rec<-c(rec,min(VariableImportance))
  #testset[IND] = NULL
  #cat("\nOverallComplexity ",overallVariableImportance)
  cat("\nMin Complexity ",min(VariableImportance))
  #cat("\ntestAUC ",testAUC[i])
  cat("\nAdding variable ", var_to_add)
 # cat("\nafterComplexity ",afterVariableImportance)
    flush.console()
}
names(x)
ncol(x)
rvar
rec
rvar[c(1:which(rec==min(rec))[1])]
F<-order(rec)
cat("\nSelected variables:", rvar[1:F[1]])
F<-order(rec)
##########################################################
#for feature selection using the backward and forward selection
##########################################################
subset<-names(X[,rvar[-c(1:which(rec==min(rec))[length(which(rec
    ==min(rec)))])]])
f <- as.simple.formula(subset, "Class")
print(f)

## for forward selection
subset<-names(X[,rvar[c(1:which(rec==min(rec))[1])]])
f <- as.simple.formula(subset, "Class")
```

```
print(f)
```

6.7.2 Results

Data sets	Methodology	Feature used	Accuracy	G Mean	Sensitivity	Specificity	AUC
WDBC	Nil	30	0.920	0.910	0.880	0.940	0.910
	CM	6	0.948	0.939	0.914	0.966	0.940
	RF(B)	30					
	GA	5	0.946	0.938	0.914	0.963	0.939
	Chi	6	0.926	0.915	0.876	0.955	0.916
	IG	6	0.926	0.915	0.876	0.955	0.916
	GR	6	0.924	0.912	0.871	0.955	0.913
	SU	6	0.924	0.913	0.876	0.952	0.914
	RReliefF	6	0.944	0.938	0.919	0.958	0.938
	PCA	7PC	0.948	0.941	0.914	0.969	0.942
Sat	Nil	37	0.920	0.741	0.573	0.957	0.765
	CM	28	0.920	0.763	0.611	0.952	0.782
	RF(B)	37					
	GA	6					
	Chi	18	0.900	0.716	0.545	0.941	0.743
	IG	18	0.890	0.702	0.526	0.938	0.732
	GR	18	0.900	0.727	0.564	0.937	0.750
	SU	18	0.900	0.743	0.588	0.940	0.764
	RReliefF	18	0.920	0.748	0.583	0.959	0.771
	PCA	8	0.880	0.730	0.578	0.921	0.749
Pima	Nil	8	0.710	0.680	0.610	0.760	0.685
	CM	5	0.726	0.700	0.634	0.774	0.704
	RF(B)	8					
	GA	3	0.638	0.552	0.404	0.764	0.584
	Chi	3	0.706	0.648	0.528	0.800	0.664
	IG	6	0.716	0.690	0.630	0.756	0.693
	GR	6	0.716	0.690	0.630	0.756	0.693
	SU	6	0.716	0.690	0.630	0.756	0.693
	RReliefF	6	0.706	0.668	0.581	0.770	0.676
	PCA	6	0.694	0.661	0.585	0.748	0.666
Abalone	Nil	8	0.880	0.430	0.200	0.940	0.570
	CM	5	0.874	0.455	0.221	0.942	0.581
	RF(B)	8					
	GA	3	0.878	0.396	0.167	0.952	0.560
	Chi	8					
	IG	8					
	GR	88					
	SU	8					
	RReliefF	7	0.870	0.435	0.205	0.938	0.571
	PCA	2	0.880	0.443	0.208	0.947	0.577
Yeast	Nil	8	0.940	0.830	0.720	0.970	0.845
	CM	2	0.942	0.834	0.719	0.970	0.845
	RF(B)	5	0.942	0.827	0.706	0.970	0.838
	GA	3	0.880	0.000	0.000	0.985	0.492
	Chi	1	0.922	0.808	0.688	0.952	0.820
	IG	3	0.942	0.806	0.669	0.973	0.821
	GR	3	0.942	0.806	0.669	0.973	0.821
	SU	3	0.942	0.806	0.669	0.973	0.821
	RReliefF	6	0.942	0.827	0.706	0.969	0.838
	PCA	6	0.922	0.770	0.625	0.955	0.790
Iono	Nil	34	0.894	0.884	0.856	0.916	0.886
	CM	6	0.918	0.910	0.880	0.942	0.911
	RF(B)	34					
	GA	5	0.862	0.828	0.752	0.920	0.836
	Chi	13	0.890	0.870	0.808	0.938	0.873
	IG	34					
	GR	34					
	SU	34					
	RReliefF	16	0.870	0.857	0.824	0.898	0.861
	PCA	5	0.638	0.592	0.488	0.724	0.606
Vehicle	Nil	18	0.780	0.690	0.560	0.850	0.705
	CM	6	0.760	0.643	0.495	0.848	0.671
	RF(B)	18					0.000
	GA	3	0.694	0.479	0.286	0.832	0.559
	Chi	10	0.760	0.667	0.533	0.837	0.685
	IG	16	0.796	0.706	0.576	0.870	0.723
	GR	16	0.796	0.706	0.576	0.870	0.723
	SU	16	0.796	0.706	0.576	0.870	0.723
	RReliefF	18					
	PCA	8	0.746	0.630	0.486	0.832	0.659
WPBC	Nil	31	0.636	0.426	0.244	0.753	0.499
	CM	16	0.646	0.558	0.444	0.707	0.576
	RF(B)	4	0.642	0.418	0.267	0.753	0.510
	GA	3	0.642	0.421	0.244	0.760	0.502
	Chi	31					
	IG	31					
	GR	31					
	SU	31					
	RReliefF	13	0.656	0.470	0.311	0.760	0.536
	PCA	9	0.688	0.561	0.422	0.767	0.594
Aba<7	Nil	8	0.920	0.750	0.590	0.960	0.775
	CM	5	0.928	0.764	0.604	0.965	0.785
	RF(B)	8					
	GA	3	0.926	0.760	0.598	0.966	0.782
	Chi	8					
	IG	3	0.924	0.733	0.557	0.968	0.763
	GR	3	0.924	0.733	0.557	0.968	0.763
	SU	3	0.924	0.733	0.557	0.968	0.763
	RReliefF	7	0.928	0.753	0.587	0.968	0.777
	PCA	3	0.926	0.764	0.609	0.963	0.786

Chapter 7

Discussion

Data Mining and Statistical/Machine learning techniques have problems handling imbalanced data sets that frequently arise in real-world applications. Researchers have developed many new techniques for tackling the class imbalance problem. Despite numerous new algorithms and re-sampling techniques, the class imbalance problem yet is to be solved. These methodologies help in solving the 'imbalance problem' in some studies, but did not help in other studies [1, 2]. Researchers like Jo and Japkowicz [3] suggested class imbalances may yield small disjunct, causing degradation, and Prati et. al. [4] argued that class imbalance in the presence of class overlap causes problems. Given the suggestion that the imbalanced class distribution may not be the only problem [5, 6, 3, 7], to the best of our knowledge there is no study that has systematically studied, what intrinsic features of the data are affecting the degraded learning performance of an imbalanced data set in real world situations.

It is important to not only design a good classifier, but to have a limit or bound on the achievable classification for a given data set. To this end, we have presented "Complexity Measurement (CM)" for systematic study of the class imbalance problem. We have focused more on tools for quantification of the problem associated with imbalance data sets rather than solving the problem, which we see as the first step for answering questions such as:

1. What is the nature of the class imbalance problem, i.e. in what situa-

tions and to what extent does class imbalance hinder the performance of standard classifiers?

2. What are the possible solutions in dealing with class imbalance problems, and how well do they perform? For example, when should we employ over-sampling or under-sampling instead of fixing a class distribution or size?

An important theoretical result related to the nature of class imbalance is presented in [8, 9]. The experiments there were conducted on artificially generated data, in the attempt to simulate different imbalance ratios, complexities and data set sizes. In general the estimated error rates from artificial data sets should be interpreted as optimistic because the analysis uses appropriate models for the data; for real data one does not know what classification methods are appropriate. Hence the important question is: how to measure data complexities in real data sets? Unless data complexities can be measured in real world data sets (a common criterion for artificial and real data sets), the conclusion drawn from experiments on synthetic data cannot be extended to real data sets.

Data complexity is not a new concept in the data mining literature. Ho and Basu [10] proposed various complexity measures for binary classification problems. But these measurements were never meant for class imbalance data sets and should be carefully interpreted. Hence application in a class imbalance scenario is debatable. This may be the reason that none of the studies [11, 12, 13] found these measurements useful, to describe the way a classifier behaves or why any re-sampling techniques does/doesn't work.

We see a need for an independent data complexity measure (CM), by which we can arrange the data sets according to their level of difficulty. By considering the level of difficulty, we can decide which learning algorithm or re-sampling technique works better. Hence CM can be used as meta-learner to decide which learning algorithm or re-sampling technique will be worth to apply and which situations need more consideration. Our study shows that, in an imbalanced problem, the CM can be used to choose the appropriate classifier, feature and re-sampling techniques that best fit the situation.

7.1 Discussion and Conclusion of Chapter 3

In this chapter our focus is to explore the connection between the imbalanced data problem and data complexity proposed in the literature [10] and our novel approach "Complexity Measurement" (CM). Our study showed that in cases of imbalance our complexity measurement (CM) is better than commonly used complexity measures in the literature [10], being within the proposed bounds of Bayes error given in the literature, such as Mahalanobis bounds and Battacharayya distance [14]. It is close to the lower bound in the case of moderate class imbalance. For severe imbalance where we choose a larger k (for kNN approach) it is even closer. Although the nearest neighbor technique is used in the literature [15] to estimate Bayes error, our contribution here is to extend it to class imbalance situations by decomposing the overall error into contributions from each class. Different distance functions can be used for determining the nearest neighbor computation. Tsymbal et. al. [16] show that Random Forest distances are better than the commonly used heterogenous Euclidean/Overlap metric (HEOM) of Wilson and Martinez [17] for the categorical or mixed type data sets. We use Euclidean distance for continuous variables, and Random Forest distance [18] for nominal variables/mixed data sets.

We have shown that CM is able to capture a reasonable amount of data complexity despite the diverse nature of data sets. We used simulated data sets to build up a complexity model and to describe different types of imbalanced data sets. Using multi-dimensional scaling (MDS) enables us to visualize the data sets and to visually interpret some of the findings of our CM. We extended the complexity analysis from synthetic data to real data sets and found a strong correlation between the sensitivity value and CM irrespective of classifier/learning algorithm used, making it an ideal meta-learner. From the existing data complexity measures proposed by Ho and Basu [10] only Fisher discriminant analysis ratio (F1) showed some correlation in our analysis of imbalanced data sets. A possible reason is that it calculates mean and variance for each class separately, hence considering the class distribution within each data set, which is important in class imbalance

situations. It is interesting to note that F1 was also found useful in feature selection [19].

We believe that our CM will be an ideal candidate to be recognized as a "goodness criterion" for various classifiers, re-sampling and feature selection techniques in the class imbalance framework. As mentioned earlier, our CM is within theoretical bounds of Bayes error, and this parameter is recognized as a benchmark for other classification techniques and also as a "goodness" criterion for feature selection used in classification [20]. This error rate will be greater than zero whenever class distributions overlap. Hence measurement of our CM, can cover the data complexity concept of class overlap [4] and small disjunct [3], more effectively than the vague concept of clustering which itself is highly dependent on k (number of nearest neighbors), to cater for the small disjunct concept.

7.2 Discussion and Conclusion of Chapter 4

Despite numerous algorithms and re-sampling techniques proposed in the last few decades to tackle imbalanced classification problems, there is no consistent winning strategy for all data sets (neither in terms of sampling, nor learning algorithm). Special attention needs to be paid to the data in hand. In doing so, one should take into account several factors simultaneously: the imbalance rate, together with data complexity, the algorithms and their associated parameters. We have used CM as a meta-learner to choose the classifier and under-sampling strategy that best fits the situation.

From the experiment it is very clear that by using CM as threshold (grouping criterion) we notice the performance of most classifiers is similar on all data sets within a particular CM group. This suggests that we don't have to use different classifiers for different CM groups; some are relatively worse than others, but not significantly (except Logistic Regression). The results for our complexity measure for different classifiers can be validated by the results for artificial data sets displayed by Figure 7.1 in which, with a range of different classifiers, for severe imbalanced data sets (minority class $< 10\%$ of the whole data set), we are getting 100% accuracy for the majority class and

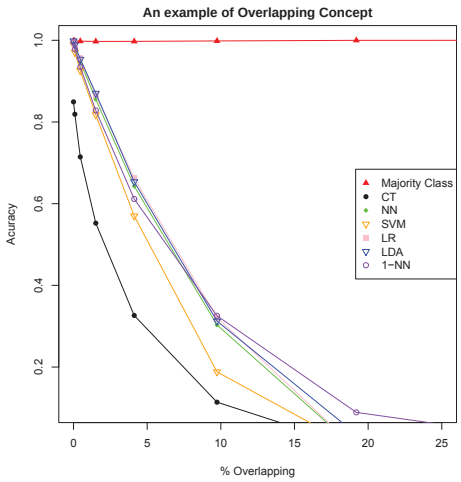

Figure 7.1: Classification for Severe Imbalance distribution. Accuracy of majority class (specificity) is shown by triangles (red) and Accuracy of minority class (sensitivity) is shown by the circle (black) over different level of overlap

no classification of the minority class in the presence of high overlap. This supports our argument that we can measure the complexity of data set more effectively in real world situations. The consistency and uniform behavior of classifiers within each CM group proves that this may be an appropriate grouping criterion.

All classification algorithms are affected to some extent by the class imbalance problem. The results on real world benchmark data indicate that the Neural Network (NN) is the most robust to the imbalance. This comes as a confirmation to the conclusions presented in [8]. On the other hand, the SVM is largely affected by the imbalance in our experiments. This contradiction with previous studies may be explained through the nature of the experimental data: on artificial data, the SVMs performed well because the systematic process of generating the data allows the existence of good support vectors (even at high imbalance ratios), whereas the existence of such vectors in real world imbalanced data sets is less probable.

In previous studies the class imbalance ratio (IR) was used as the threshold to describe the level of difficulty [21, 22]. But our experiments prove that

IR is not an adequate threshold variable to describe the effectiveness of any preprocessing techniques.

For imbalanced data-sets with CM < 50, under-sampling techniques can produce acceptable results for the minority class (i.e., accuracy of minority class $> 80\%$) without losing much accuracy for the majority class. But for imbalanced data sets with CM> 50, which needs much greater consideration, techniques that worked for lower CM values were not effective in this region. The balancing act of training data sets severely under-sampled the majority class, which resulted in losing its accuracy by around 30% and resulted in specificity lower than sensitivity. For this CM region we recommend a combination of over-sampling and under-sampling techniques, which may help in gaining better accuracy for both the majority and minority class. In Chapter 5 we have shown that in the overlapped regions over-sampling works better than just under-sampling.

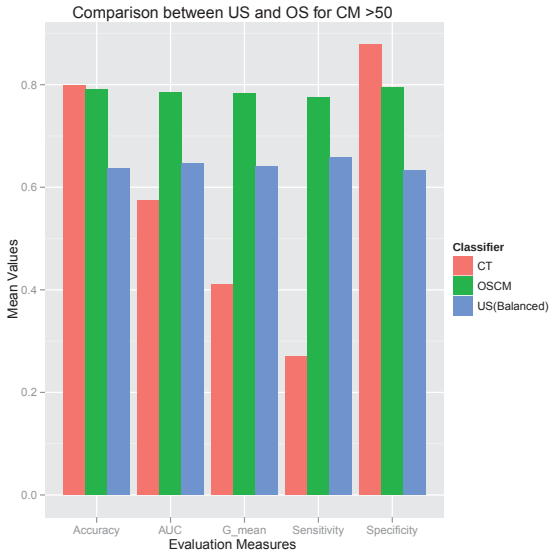

Figure 7.2: Comparison between balanced under-sample and Over-sampling using CM for CM > 50

We have applied our proposed OSCM (over-sampling using complexity measurement) on all the data sets with CM > 50. Relative performance is

displayed in Fig. 7.2, where we can see that OSCM is better than balanced under-sampling of the training set for all evaluation measures for more complex data sets i.e., CM > 50. Hence it can be concluded that for imbalanced data sets with high complexity, random over-sampling works better than random under-sampling techniques. But a combination of over-sampling and under-sampling potentially creates ideal classifiers [7, 23]. A combination of these two methodologies will be considered in our future work.

7.3 Discussion and Conclusion of Chapter 5

It is well known that representativeness in the training data is the most important step to having a classifier with high generalization performance [5]. However in most classification problems, representation is not only costly but a very cumbersome task. In general data sets are small, sparse, with missing values and with highly imbalanced prior probabilities.

In order to address the problem and to bring some structure to over-sampling within the data sets, we have proposed two new methods of over-sampling: synthetic over-sampling technique using complexity measure and random over-sampling using complexity measure. Experimental results for several imbalanced data sets have indicated that the proposed algorithms can result in better prediction of minority class, whereas prediction performance of majority class does not suffer much. Data sets used in our experiments contained different degrees of imbalance, different sizes and different data complexities (as shown by CM value).

Introducing complexity measure (CM) as a pre-screening tool helps us to identify difficult examples belonging to minority class, and simple over-sampling or SMOTE algorithm improves the performance of a classifier on these minority class examples. Therefore, over-sampling these difficult examples forces the classifier to focus more on the difficult examples that belong to minority class rather than to the majority class. Unlike standard boosting where all misclassified examples are given equal weights, our approach creates examples from the minority class (by SMOTE or over-sampling), thus indirectly changing the updating weights and compensating for imbalanced

distributions. Moreover this is not a wrapper technique like boosting, so the resultant training set can be used with any classifier. By introducing over-sampling proportional to the complexity of data sets, we were able to dictate the amount of SMOTE or over-sampling within a data set. The appropriate amount of over/under sampling required is still an open question in the data mining community [24, 25, 26]. Unlike existing methods like random over-sampling, SMOTE [23] or BSM [27], we do not need to search for an optimal class distribution in order to get favorable accuracy of the minority class since the amount of over-sampling is determine by the complexity. Experiment has indicated that our proposal leads to better behavior (high accuracy of minority class), which validates the efficiency of our method.

By basing our complexity measure on k-NN where $k > 2$ we are able to identify the minority class points near the decision boundary. This information can be useful in interactively changing some important aspect of a data set like intra vs inter class imbalance and small disjunct [3] or class overlap [4]. Using CM, we can identify and select points near to the decision boundary, then see the effect it can produce on the decision boundary when we over-sample these data points by SMOTE or by random over-sampling technique, which leads to better accuracy of the minority class.

7.4 Discussion and Conclusion of Chapter 6

Feature selection for highly imbalanced data is a major challenge in Data Mining and Statistics. As mentioned earlier, our CM can be used as the "goodness criterion", so we have proposed a sequential feature selection method using complexity measure (CM) to focus on the class imbalance problem. Feature selection is a relatively new technique to tackle class imbalance problems. To evaluate the effectiveness of the proposed method, we have applied our technique on well known UCI data sets from the class imbalance literature. Using the CM criterion which is independent of the classifier, selected features can be used with different classifiers to build a number of classification models. We have also compared the new method to existing techniques/approaches. The experimental results demonstrate that (1) our

proposed method performed significantly and consistently better than the other approaches; and (2) the performance strength of our proposed method over other methods becomes increasingly evident as the class imbalance becomes more serious. The results also strengthen the claim that this parameter (CM) should be recognized as a benchmark for other classification techniques and also as a "goodness" criterion for feature selection used in classification.

7.5 Large data sets

The data sets used in our investigation have been of modest size by data mining standards. The question arises of the feasibility and performance of the methods on large data sets. For brief illustration with large data sets, we use two real data set (Motor Insurance and Forest Cover Type), description of which is given in Appendix 7.7. For the Forest Cover Type data set, 10 quantitative variables were used. The binary class target of two classes is formed by the majority species Spruce-Fir with 211840 observations denoted by class '0', whereas the minority class is formed by combining two classes Cottonwood/Willow (2747) and Aspen (9493) denoted by class '1'. Hence we get a majority class with 211840 (94.5%) observations and minority class with 12240 (5.5%) observations. Motivation behind this analysis is to select two large data sets with similar degree of imbalance and to exhibit their different natures. Our CM gives 100% (i.e., all examples in the minority class are classified as difficult to learn) for the Motor Insurance data set, hence showing complete overlap for minority class, whereas CM for the Forest data set is at around 9%.

Table 7.1 contains the specificity and sensitivity for the prediction computed from the test data (40% of whole data set). It is clear from the results that none of the classification techniques is able to predict the minority class for Motor Insurance, even though the minority class has approximately the same number of observations i.e. 14701 in comparison to Forest Cover Type data set. This might be indicating that there is large overlap between the classes or non-separable classes, as shown by our CM values. Whereas for the Forest Cover Type data set, it is clear that all of the classification techniques

are able to predict the minority class, with fairly high accuracy. Although the minority class has approximately the same proportion of observations as that of Motor Insurance data, this data set has a low CM value.

Table 7.1: Confusion Matrices for the Motor Insurance Data set (No Claims: 285,299 (95.1%); Claims: 14,701 (4.9%)) and Forest Cover Type Data set (Spruce-Fir('0'): 211,840 (94.5%); Cottonwood/Willow and Aspen ('1'), 12,240 (5.5%))

Classification Model	Motor Insurance		Forest Cover Type	
	No claims (0) Specificity	claims (1) Sensitivity	Class(0) Specificity	Class (1) Sensitivity
Classification Tree	1	0	99.44	87.85
Random Forest	1	0	99.91	94.91
Neural Network	1	0	99.37	90.54
Support Vector Machine	1	0	98.51	95.28

Forest Cover Type data has high accuracy from all classifiers, hence will not be considered for further processing. Since all classification techniques are unable to predict the minority class for Motor Insurance, we want to examine iterative sampling techniques to help classifying the minority class. We divide the entire majority class into 19 random partitions to get roughly balanced training sets, and conduct 10 fold cross validation. Table 7.2 shows the mean (variance) of specificity and sensitivity for the test set.

Table 7.2: Confusion Matrix for the Motor Insurance Data sets: Roughly Balanced Random Partition (No Claims:15015 (50.5%);Claims: 14,701 (49.5%))

Classification Model	No claims (0)	claims (1)
	Specificity	Sensitivity
Classification Tree	55.2 (1.04)	57.3 (0.63)
Random Forest	51.5 (0.54)	63.7 (0.18)
Neural Network	38.8 (2.22)	77.9 (1.55)
Support Vector Machine	62.5 (1.02)	57.1 (0.22)

As expected Table 2 shows that all the classification techniques are now able to predict the minority class but at the expense of accuracy of majority class. Since CM is very high for this data set, it is a compromise between

sensitivity and specificity. None of the techniques proves to be different from the others. This preliminary study suggest that data characteristics remain the same for comparatively large data sets.

7.6 Conclusions and Future Work:

In this work we have analyzed the classifier and preprocessing effect in the framework of imbalanced data sets by means of data complexity. We present a Complexity Measurement (CM) for systematic study of class imbalance problems. CM can be be used in two ways: to measure the level of difficulty in imbalanced data sets and as a tool to help cater for the problem. The major challenge in analyzing highly imbalanced data is that usually a small number of minority class are mixed with an overwhelming majority class. We have tackled almost every aspect that we think is critical to making a good prediction for a class-imbalanced data set, including choices of classifier, feature selection and both up and down sampling techniques when generating a training set.

The result in the thesis is that CM can be used as a meta-learner in order to choose the classifiers and sampling techniques that best fit the situation. We have observed that Imbalance Ratio (IR) and complexity measure proposed by Ho and Basu [10] considered as measures of data complexity are not enough to predict when a particular classifier or re-sampling technique will perform well or badly. As an alternative we introduce CM for imbalanced data sets in order to obtain ranges of complexity in which classifier, feature selection and re-sampling techniques perform well or badly. From these ranges or groups we draw conclusions which have wide support and show a significance difference with respect to the global methods of performance (across all data sets within range of complexity).

We have finally obtained two groups from the initial ones (i.e., CM < 50 and CM > 50), which are simple and precise to predict either good or bad performance of the classifier and re-sampling techniques. These two groups are capable of identifying good or bad data sets for re-sampling techniques. An interesting consequence of the characterization obtained by these groups

is that we can select re-sampling techniques capable of preprocessing data sets more successfully for any classifier. These two groups may be taken as a judging criterion for new algorithms and re-sampling techniques. Moreover, new data sets may be evaluated using these groups in order to build a sensible model.

The use of a cost-sensitive classifier is another way of addressing the class-imbalance problem [28]. A cost-sensitive approach assign a higher cost to a misclassified minority class than to a false alarm of a majority class, to enforce the priority of predicting minority class. The methodology introduced in Chapter 5, of adding data proportional to its complexity when applying the off-the-shelf methods, could in fact be regarded as a cost-sensitive classifier. However we observed that with just cost reassignment, the resulting classifier is not as capable as our proposed method. In OSCM, the over-sampling technique can to some extent be comparable to the idea of cost-sensitivity; the difference is that the over-sampling adjusts the class-sizes so that misclassification cost directly applied to the data is not needed. For a classifier to work well for highly imbalanced data especially for data sets where CM $>$ 50, OSCM recommended by our research seem to be critical and necessary.

Existing feature selection techniques are not adequate for imbalance data sets. These feature selection techniques were never meant for class imbalance problems, hence it was deemed necessary to devise a strategy/technique to select features that work in the class imbalance scenario. We have proposed a novel technique, Feature Selection using Complexity Measurement (FSCM), which works more efficiently and consistently across different data sets. We were able to increase the accuracy of the minority class without compromising the accuracy of majority class. Complexity Measurement (CM) is adequate evaluation criterion for wrapper methodology, which was lacking in the literature. CM gives error estimation for the particular class, independent of the classifier used.

Ultimately we will say that if someone has the task of classifying imbalanced data sets, we strongly advise to measure the complexity measurement CM of the data, which will give an idea of level of difficulty in class learning. Such an estimate will help researchers/users to decide whether to improve

the current classifier, to use another classifier on the same data set or to acquire more data. We will definitely recommend our proposed method like OSCM and FSCM in order to increase the accuracy of the minority class, without seriously compromising the accuracy of majority class. Over all, we can say that there is no well established, proven, method for generally handling class imbalance and our research is to develop an approach which works for most cases. We proposes a framework Complexity Measurement (CM) for systematic study of class Imbalance Problems. CM can be used in two way: to measure the level of difficulty in imbalanced data sets and as a tool to help cater for the problem.

In our future work we intend to make our proposed method available in an R package (Important R codes/functions are already provided in the Appendices), which can be applied to class-imbalance problems that often appear in real-world data mining. We want to investigate using complexity measure as an under-sampling scheme, to eliminate noisy examples. If we SMOTE between noisy examples, will it lead to driving the noise towards the correct labeled examples? In the presence of mislabeling of positive class examples as negative class examples, we expect complexity measurement would be able to identify these examples so that we can eliminate them and force the classifier to focus more on the minority class. Although a new classifier for imbalanced data sets is not yet feasible, a meta-learner tool such as we advocate in this Thesis should be applicable in improving the accuracy of minority class. Despite the success of proposed complexity measurement (CM) in efficiently quantifying the level of difficulty of imbalanced data sets, there is still much room (a great need as well) to further improve. We need efficient re-sampling and feature selection techniques to improve accuracy of minority class especially for the group CM>50, via innovative future research.

References

[1] Kubat, M., and Matwin, S. Addressing the Curse of Imbalanced Data Sets: One Sided Sampling. In the Proceedings of the Fourteenth International Conference on Machine Learning, Nashville, pp. 179-186, 1997.

[2] Drummond, C. and Holte, R., Severe Class Imbalance: Why Better Algorithms Arent the Answer. In Proceedings of the 16th European Conference on Machine Learning (ECML/PKDD'05), pp. 539-546, 2005.

[3] Jo, T., Japkowicz, N. Class imbalances versus small disjuncts. SIGKDD Explorations, Vol. 6, pp.429-450, 2004.

[4] Prati, R. C., Batisita, G. E., and Monard, M. C., Class Imbalances versus Class Overalpping, An Analysis of the Learning System Behaviour, in MICAI, pp. 312-321, 2004.

[5] Weiss, G.M., Mining with Rarity: A Unifying Framework. SIGKDD Explorations, Vol. 6, no. 1, pp. 7-19, 2004.

[6] Prati, R. C., Batista, G. E. A. P. A., and Monard, M. C., Class imbalances versus class overlapping: an analysis of learning system behavior. In MICAI, pp. 312-321, 2004.

[7] Provost, F., and Fawcett, T., Robust classification for imprecise environments, Machine Learning Journal, Vol. 42, pp. 203-231, 2001.

[8] Japkowicz N. and Stephen, S.,"The Class Imbalance Problem: A Systematic Study", Intelligent Data Analysis, Vol. 6, no. 5, pp. 429-450, 2002.

[9] Garcia, V., Mollineda, R., Sanchez, J., S., On the k-NN performance in challenging scenario of imbalance and overlapping. Pattern Analysis Application, Vol. 11, no. 3-4, pp. 269-280, 2008.

[10] Ho, T. K., Basu. M, Complexity measures of a supervised classification problems, IEEE Trans Pattern Analysis, Machine learning, Vol. 24, no. 3, pp. 289-300 , 2002.

[11] Weng, C. G. and Poon, J., A data complexity anylysis on imbalanced datasets and an alternative imbalance recovering strategy, Proceeding of the IEEE/WIC/ACM International Conference on Web Intelligent, 2006.

[12] Li, Y., Member, S., Dong, M., Kothari, R., Classificability-based omnivariate decision trees, IEEE Trans Neural Netwrok, Vol. 16, no. 6, pp. 1547-1560, 2005.

[13] Bernado-Mansilla, E., Ho, T., K., Domain of competence of XCS classifier system in complexity measurement space. IEEE Trans Evol Computation, Vol. 9, no. 1, pp. 82-104 , 2005.

[14] Fukunga, K., Introduction to Statistical Pattern Recognition, 2nd edition. Boston: Academic Press, 1990.

[15] Cover, T. M., and Hart P. E., Nearest Neighbor pattern classification. IEEE Transactions on Information Theory, Vol. 13, pp. 21-27, 1967.

[16] Tsymbal, A., Pechenizkiy M., Cunningham P., Dynamic integration with random forests, in: ECML06, 17th European Conference on Machine Learning, Berlin, Germany (2006), pp. 801808 (extended version is available online at http://www.cs.tcd.ie/publications/tech-reports/reports.06/TCD-CS-2006-23.pdf).

[17] Wilson D.R., Martinez T.R. Improved heterogeneous distance functions. Journal of Artificial Intelligence Research, Vol. 6, no. 1, pp. 1-34, 1997.

[18] Breiman, L., Random Forest, Machine Learning, Vol. 45, no. 1, pp. 5-32, 2001.

[19] Mollineda, R. A., Sanchez, J. S. and Sotoca, J. M., Data characterization for effective prototype selection. In: First edition of the Iberian conference on pattern recognition and image analysis, Lecture Notes in Computer Science 3523, pp. 27-34, 2005.

[20] Fukunga, K., and Krile, T. F., Calculation of Bayes Recognition Error for Two Multivariate Gaussian Distributions, IEEE Transaction on Computer, Vol. 18, no. 3, pp. 220-229, 1969.

[21] Hulse, J.V., Khoshgoftaar, T.M., Napolitano, A., Experimental perspectives on learning from imbalanced data, in: Proceedings of the 24th International Conference on Machine Learning, Corvalis, Oregon, pp. 935-942, 2007.

[22] Batista,G.E., Prati, R.C., and Monard,M.C., Balancing strategies and class overlapping. In proceeding of 6th International Symposium on Intelligent Data Analysis, Madrid, Spain, pp. 24-35, 2005.

[23] Chawla, N., Bowyer, K.,Hall, K., Kegelmeyer, P., SMOTE: Synthetic Minority Over-Sampling Technique, Journal of Artificial Intelligence Research, Vol.16, pp. 321-357, 2002.

[24] Estabrooks, A., Jo, T., and Japkowicz, N., A Multitple Resampling Method For Learning From Imabalanced Data Sets. Computational Intelligence, Vol. 20, no. 1, pp. 18-36, 2004.

[25] Chan, P.K. and Stolfo, S.J., Towards Scalable learning with non-uniform class and cost distribution. A case study in credit card fraud detection, in proceeding of the foruth Conference on Knowledge Discovery and Data Mining, New York,, pp. 164-168, 1998.

[26] Hall, L. O., Joshi, A., Building Accurate Classifiers from Imbalanced Data Sets, IMACS05, Paris, 2005.

[27] Han, H., Wang, W., Mao, B., Border-SMOTE: A New Over-Sampling in Imbalanced Data Sets Learning. LNCS Springer, Heidelberg, pp. 878-887, 2005.

[28] Domingos, P., MetaCost: A general method for making classifier cost sensitive, in Proceeding of the Fifth International Conference on Knowledge Discovery and Data mining, ACM Press, pp. 155-164, 1999.

7.7 Appendix

Motor Insurance Data set

The Motor insurance data shows records of 300,000 claims from a UK comprehensive motor portfolio i.e. for standard full cover car insurance. The only real difference in cover for NZ compared to the UK is that ACC picks up any personal-injury claims. The data set shows the number of claims (fields include "CLMS" in their names), and their incurred cost (fields include "INCUR" in their names) in the following categories:

- WS = windshield
- AD = accidental damage (to third parties)
- FT = fire and theft
- PD = personal damage
- PI = personal injury

The "FROMDATE" and "TODATE" give the period of time over which each policy was insured with the characteristics described by the other fields on the record. for this study it have been changed to Duration which indicates number of days for each policy insured. The number of attributes used for this study are USAGE,MILEAGE,SEX ,EXCESS,PRIMAGE, MINAGE, DRIVERS, (FROMDATE and TODATE) convert to Duration, CARGROUP, CAR AGE, (WSCLMS, ADCLMS, FTCLMS, PDCLMS and PICLMS) convert to binary target variable claims, RECORD EXPOSURE and

DURATION of the original 27 attributes. The claims is our target variable with number of non claimant"1" is 285,299 (95.1%) and number of claimant is 14,701 (4.9%).

Forest cover type data set

A large data set, forest cover type data set from UCI KDD archive was used. The following seven forest cover type classes used in a classification problem were:

- C1: Spruce/fir (Picea engelmannii and Abies laciocarpa),
- C2: Lodgepole pine (Pinus contorta),
- C3: Ponderosa pine (Pinus ponderosa),
- C4: Cottonwood/willow (Populus angustifolia, Populus deltoides, Salix bebbiana, Salix amygdaloides),
- C5: Aspen (Populus tremuloides),
- C6: Douglas-fir (Pseudotsuga menziesii),
- C7: Krummholz (Engelmann spruce (Picea engelmannii), subalpine fir (Abies lasiocarpa) and Rocky Mountain bristlecone pine (Pinus aristata)).

The forest cover type for the 30-30m cells obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data represents the primary dominant tree species currently found in four wilderness areas (Neota with 3904 ha, Rawah with 29628 ha, Comanche Peak with 27389 ha, Cache la Poudre with 3817 ha) located in the Roosevelt National Forest of northern Colorado. In these areas, the existing forest cover types are primarily a result of natural ecological processes which experienced relatively little human disturbances in the past. Some interesting background information for these four wilderness areas are the mean elevational value and the primary tree species found there. Neota area has the highest mean elevational value and a primary major tree species spruce/fir (C1). Rawah and Comanche Peak areas would have a lower mean elevational value and lodgepole pine

(C2) as their primary tree species, followed by spruce/fir (C1) and aspen (C5). Cache la Poudre areawould have the lowest mean elevational value and Ponderosa pine (C3), Douglas-fir (C6) and Cottonwood/willow (C4). From US Geological Survey (USGS) and USFS original data, a total number of 581,012 observations with 54 attributes was defined in the cover type data set (UCIKDD archive, 2005). The distribution of seven data classes is presented as class C1 with 211,840 observations, class C2 with 283,301 observations, class C3 with 35,754 observations, class C4 with 2747 observations, class C5 with 9493 observations, class C6 with 17,367 observations and class C7 with 20,510 observations. The forest cover type data set was treated as a multi-class classification problem with imbalanced number of observations in the last five classes. The total number of 54 attributes of cover type data availably include the following 12 measures defined with 10 independent quantitative variables, four binary wilderness areas and forty binary soil type variables. The quantitative variables are:

- Elevation in range 1859.3858 (m).
- Aspect (azimuth from true north) in range 0.360 (azimuth).
- Slope in range 0.66 (.).
- Horizontal Distance To Hydrology in range 0.1397 (m).
- Vertical Distance To Hydrology in range .173.601 (m).
- Horizontal Distance To Roadways in range 0.7117 (m).
- Hillshade 9 a.m. in range 0.255 (index).
- Hillshade Noon in range 0.255 (index).
- Hillshade 3 p.m. in range 0.255 (index).
- Horizontal Distance To Fire Points in range 0.7173 (m).

In observations, the wilderness area designation and soil type information are defined with binary variables treated as multiple binary values where

a value '0' would represent an absence and a value '1' would represent a presence of a specific wilderness area or soil type.