Combating the Class Imbalance Problem in Small Sample Data Sets

Michael Wasikowski

Submitted to the Department of Electrical Engineering & Computer Science and the Graduate Faculty of the University of Kansas School of Engineering in partial fulfillment of the requirements for the degree of Master's of Science

Thesis Committee:

Dr. Xue-wen Chen: Chairperson

Dr. Jun Huan

Dr. Brian Potetz

Date Defended

The Thesis Committee for Michael Wasikowski certifies
That this is the approved version of the following thesis:

Combating the Class Imbalance Problem in Small Sample Data Sets

Committee:

Chairperson: Dr. Xue-wen Chen

Dr. Jun Huan

Dr. Brian Potetz

Date Approved

Abstract

The class imbalance problem is a recent development in machine learning. It is frequently encountered when using a classifier to generalize on real-world application data sets, and it causes a classifier to perform sub-optimally. Researchers have rigorously studied resampling methods, new algorithms, and feature selection methods, but no studies have been conducted to understand how well these methods combat the class imbalance problem. In particular, feature selection has been rarely studied outside of text classification problems. Additionally, no studies have looked at the additional problem of learning from small samples. This paper develops a new feature selection metric, Feature Assessment by Sliding Thresholds (FAST), specifically designed to handle small sample imbalanced data sets. FAST is based on the area under the receiver operating characteristic (AUC) generated by moving the decision boundary of a single feature classifier with thresholds placed using an even-bin distribution. This paper also presents a first systematic comparison of the three types of methods developed for imbalanced data classification problems and of seven feature selection metrics evaluated on small sample data sets from different applications. We evaluated the performance of these metrics using AUC and area under the P-R curve (PRC). We compared each metric on the average performance across all problems and on the likelihood of a metric yielding the best performance on a specific problem. We examined the performance of these metrics inside each problem domain. Finally, we evaluated the efficacy of these metrics to see which perform best across algorithms. Our results showed that signal-to-noise correlation coefficient (S2N) and FAST are great candidates for feature selection in most applications.

Keywords: Class imbalance problem, feature evaluation and selection, machine learning, pattern recognition, bioinformatics, text mining.

Acknowledgments

I would like to thank Professor Xue-wen Chen for the advice and encouragement he has given me over the past two years. My intellectual growth would not have been possible without his support. Thanks also go to Professors Jun ”Luke” Huan and Brian Potetz for serving on my committee.

Thanks to the EECS department, ITTC, Mr. Leroy ”Jack” Jackson, and the SMART Program for supporting me and my research over the past two years. I'd like to thank Major Paul Evangelista for mentoring me over the past year and helping me with my upcoming transition from the university to working for the government.

I would like to thank Oscar Luaces for his assistance in compiling the MEX interfaces for SVM^{perf} which allowed me to use the most modern software in my experiments.

I would especially like to thank my parents. They showed me unconditional love and support for everything I took on as a kid and an adult, and I am eternally grateful for this. My father also instilled in me a love for computers and programming that continues to this day, and I would not be where I am today without this influence as a child.

Most of all, I would like to thank my wife, Whitney. There were times that I doubted myself and whether I would be successful, and she was the source of my strength and perseverance through these times. I cannot possibly thank her enough for everything she have given me.

A portion of this work was supported by the US National Science Foundation Award IIS-0644366. Any of the opinions, findings, or conclusions in this report are those of the author and do reflect the views of the National Science Foundation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The class imbalance problem is a difficult challenge faced by machine learning and data mining, and it has attracted a significant amount of research in the last ten years. A classifier affected by the class imbalance problem for a specific data set would see strong accuracy overall but very poor performance on the minority class. This problem can appear in two different types of data sets:

1. Binary problems where one of the two classes is comprised of considerably more samples than the other, and

2. Multi-class problems where each class only contains a tiny fraction of the samples and we use one-versus-rest classifiers.

Data sets meeting one of the two above criteria have different misclassification costs for the different classes. The costs for classifying samples into different classes are listed in a cost matrix. The specific cost matrix for a problem is occasionally explicitly stated, but much of the time, it is simply an implicit part of the problem. Thus, an algorithm will either have to determine the best cost matrix while training [59], or the user will have to select a cost matrix to use in training. If the chosen cost matrix is incorrect, it can lead to flawed decisions from the classifier, so it is extremely important when doing cost-sensitive learning that the proper cost matrix be used [27]. Cost-sensitive learning is described in depth in Section 2.2.3.

1.1 Motivation

There are a large number of real-world applications that give rise to data sets with an imbalance between the classes. Examples of these kinds of applications include medical diagnosis, biological data analysis, text classification, image classification, web site clustering, fraud detection, risk management, and automatic target recognition, among many others.

The skew of an imbalanced data set can be severe. In small sample data sets, such as those with hundreds of samples or less, the skew can reach 1 minority sample to 10 or 20 majority samples. In larger data sets that contain multiple thousands of samples, the skew may be even larger; some data sets have a skew of 1 minority sample to 100, 1000, and even 10000 majority samples, and sometimes worse. As the skew increases, performance noticeably drops on the minority class.

Why is the class imbalance problem so prevalent and difficult to overcome? Standard algorithms make one key assumption that causes this problem: a classifier's goal is to maximize the accuracy of its predictions. This is not technically correct because most modern classifiers try to optimize a specific loss function on the training data. There are many examples: regression functions attempt to minimize the least squares error of the system, the support vector machine (SVM) tries to minimize regularized hinge loss, the naïve Bayes classifier maximizes posterior probability, decision trees minimize the conditional entropy of leaf nodes while also minimizing the number of branches, and the nearest neighbor minimizes the distance of test samples to training samples. The one constant between these loss functions is that they generalize very well to overall predictive accuracy on training data. Thus, while it's not necessarily the stated goal for using a given classifier, it's implied that a classifier tries to maximize the accuracy of its predictions [41].

Based on this assumption, a classifier will almost always produce poor results on an imbalanced data set. This happens because induction algorithms have trouble beating the trivial majority classifier on a skewed data set [32]. A classifier that attempts to classify minority samples correctly will very likely see a significant reduction in accuracy [41] which tells us that the accuracy of the classifier is under-representing the value of classification on the minority class [32]. For example, consider a data set where 99% of the samples are in one class. The trivial majority classifier can achieve 99% accuracy on the data set, so unless an algorithm can beat 99% accuracy, its results will be worse than simply choosing the majority class. Thus, the interesting results arise in the accuracy scores above the majority ratio.

In most cases of imbalanced distributions, we would prefer a classifier that performs well on the minority class even at the expense of reduced performance on the majority class. Researchers use statistics like the F-measure [32, 78] and area under the receiver operating characteristic (AUC), [41] to better evaluate minority class performance. The F-measure explicitly examines the performance of the classifier on the minority class. The AUC measures the overall goodness of a classifier across all possible discrimination thresholds between the two classes. A thorough discussion of various evaluation statistics used on imbalanced data can be found in Section 4.2.

1.2 Approaches

Researchers have crafted many techniques to combat the class imbalance problem. These methods fall into one of three main types of approaches:

1. Resampling Methods
2. New Algorithms
3. Feature Selection Methods

Resampling methods strategically remove majority samples and/or add minority samples to an imbalanced data set to bring the distribution of the data set closer to the optimal distribution. New algorithms approach imbalanced problems differently than standard machine learning algorithms; some examples include one-class learners, bagging and boosting methods, cost-sensitive learners, and algorithms that maximize statistics other than accuracy. Feature selection methods select a small subset of the original feature set to reduce the dimensionality of the data set and facilitate better generalization of training samples.

1.3 Related Issues

With the explosion of information and computing power available in the last few decades, researchers have found a number of data sets with one of two issues: a large number of samples with small feature sets, and a large feature set with very few samples. The former issue can be solved in machine learning simply by adding more computing power to the algorithm. However, small samples with large feature sets are another significant problem for machine learning.

Induction algorithms need a sufficient amount of data to make generalizations about the distribution of samples. Without a large training set, a classifier may not generalize characteristics of the data; the classifier could also overfit the training data and be misled on test points [44]. Some of the different methods used to combat the class imbalance problem could make the problems with learning on a small data set even worse. In fact, Forman [33] compared the naïve Bayes and linear SVM algorithms on a number of small sample text classification problems. He found with very skewed small samples, the best performance is typically achieved by the naïve Bayes and multinomial naïve Bayes algorithms; the traditionally powerful linear SVM had rather poor performance in comparison. When we use only marginally skewed data sets, the linear SVM performs best.

There is only a small volume of research on learning from small samples, but there are a number of problem domains who would benefit greatly from research into this task. Biological data analysis problems frequently have very small sample sizes but large feature sets. This report covers nine different biological data analysis sets, including four microarray data sets and five mass spectrometry data sets. The largest of these data sets has just over 250 samples, but each data set has upwards of 7000 features for each sample. It is expensive to sequence a person's genome or analyze a person's serum for protein markers. A learning method that can use small samples but still make strong generalizations about test observations would likely save a biological researcher money on obtaining more data.

1.4 My Contribution

This thesis contains two main contributions to the learning community. I developed a new feature selection metric, Feature Assessment by Sliding Thresholds (FAST). I also conducted the first systematic study of methods from each of the three types of approaches on a number of small sample imbalanced data sets.

Previously developed feature selection methods were designed without regard for how the class distribution would affect the learning task. Thus, the use of many of them result in only moderately improved performance. In contrast, FAST was developed with the goal of achieving strong performance on imbalanced data sets. FAST evaluates features by the AUC; this is one of the most common ways to evaluate classifiers trained on imbalanced data, so it stands to reason that it would be a strong way to evaluate the features of an imbalanced data set as well.

The newest of the techniques to resolving the class imbalance problem is feature selection. Most research on feature selection metrics has focused on text classification [32, 62, 78]. There are many other applications in which it would be advantageous to investigate feature selection's performance. We will look at the performance of different feature selection metrics on microarray, mass spectrometry, text mining, and character recognition applications. We aim to inform data mining practitioners which feature selection metrics would be worthwhile to try and which they should not consider using.

Very little research has been conducted to evaluate how the different types of approaches work compared to one another on the same data sets; most of the work focuses exclusively on different methods within one type of approach. Van Hulse, Khoshgoftaar, and Napolitano examined seven different resampling methods [40], Forman surveyed twelve different feature selection methods [32], and most of the papers covering new algorithms looked at existing algorithms for performance comparison [46, 58, 59]. This report covers the performance of various resampling methods, algorithms, and feature selection methods on a number of real-world problems.

1.5 Thesis Structure

This thesis is divided into six chapters. Following the introduction to the material in Chapter 1, Chapter 2 presents background information concerning methods designed to combat the class imbalance problem, including a number of resampling methods, new algorithms, and feature selection. Chapter 3 explains the details of the various methods we used in our experiments on homogeneous data sets with Chapter 4 rigorously defining the scientific questions we aim to answer, as well as how we will answer them. Chapter 5 follows with the results of these experiments. Finally, Chapter 6 ends our report with our concluding remarks and some goals for future research on the class imbalance problem.

Chapter 2

Imbalanced Data Approaches

The two Learning from Imbalanced Data Sets workshops thoroughly explored the three different types approaches to combating the class imbalance problem: resampling methods, new algorithms, and feature selection methods. The first was held at the AAAI conference in 2000 [41], and the second was held at the ICML conference in 2003 [11]. Also, Weiss reviewed these approaches in SIGKDD Explorations [71], and Chawla, Japkowicz, and Kolcz [13] published an editorial on the history of research on imbalanced data. The vast majority of this research has so far focused on resampling methods, with new algorithms receiving a small amount of research solely on imbalanced data, and feature selection receiving the least of all.

Much of the research on combating the class imbalance problem has focused on large data sets with many thousands of samples. This is because larger data sets can have more severe imbalances; it is commonly accepted that the most difficult data sets to learn are those with the most disparate class sizes. For example, consider two data sets of different size, one with only 100 samples, and one with 10,000 samples. If we limit ourselves to a minimum of 10 minority samples for generalization purposes, then the small data set can only reach a class ratio of 1:9, but the large data set could reach a skew of 1:999, or nearly 0.1%. However, there are a significant number of domains where data sets are small but still imbalanced, and there is very little research on how to attack these kinds of imbalanced data sets.

2.1 Resampling Methods

Resampling techniques aim to correct problems with the distribution of a data set. Weiss and Provost noted that the original distribution of samples is sometimes not the optimal distribution to use for a given classifier [72]; with very imbalanced data sets, the original distribution is almost always not the best distribution to use as evidenced by the trivial majority classifier. Better class distributions will improve the validation and testing results of the classifier. Although there is no real way to know the best distribution for a problem, resampling methods modify the distribution to one that is closer to the optimal distribution based on various heuristics.

2.1.1 Natural Resampling

One simple resampling technique is to obtain more samples from the minority class for inclusion in the data set. This will help relieve the skew in the data set, and there is the added benefit that all of the samples in the data set remain drawn from the natural phenomenon that built it. However, this is not always possible in real-world applications. In many problem domains, the imbalance is an inherent part of the data [13]. The vast majority of credit card transactions conducted every day are legitimate, so to collect data about the same number of fraudulent purchases as can be collected for legitimate uses over a one day time span, we would likely have to spend a number of months, if not years, to do so. For the years 2001-2005, the incidence rate, or new cases per total people, of the fifteen most common cancers combined in the United States of America was just under 1,000 per 100,000 people, or 1% as a ratio [43], so finding the same number of cancer patients as non-cancer patients is difficult. Much of the time, the cost of the data gathering procedure limits the number of samples we can collect for use in a data set and results in an artificial imbalance [13]. For example, sequencing a person's genome requires expensive equipment, so it may not be feasible to include a large number of samples on a microarray. Many machine learning researchers simply find that you are limited to the data you have [41]. This restricts a lot of researchers to combating the small sample problem and the class imbalance problem at the same time.

2.1.2 Artificial Resampling

Other resampling techniques involve artificially resampling the data set. This can be accomplished by under-sampling the majority class [16, 51], over-sampling the minority class [10, 52], or by combining over and under-sampling techniques in a systematic manner [30]. The end result is a data set that has a more balanced distribution. Because the optimal distribution of the data is unknown, a number of researchers design their methods to fully balance the distribution so that each class has an equal number of members. Other researchers use parameterized methods that can adjust the distribution to any skew possible and then compare the results using a validation scheme.

The two most common resampling techniques are the two most simple methods: random majority under-sampling, and random minority over-sampling. The random majority under-sampling algorithm discards samples from the majority class randomly, and the random minority over-sampling method duplicates samples from the minority class randomly. Based on the findings by Elkan in his review of cost-sensitive learning [27], learning using the over-sampling method to a fully balanced distribution is quite similar to applying a cost matrix with errors having cost equal to the class ratio; the true costs when using over-sampling can differ by small amounts. These methods can be parameterized to set the class distribution to any ratio.

Kubat and Matwin [51] created a method called one-sided sampling. They identified four different types of majority samples: samples that are mislabeled because of the effect of class label noise, samples on or close to the decision boundary, samples that are redundant and contribute nothing to the learning task, and safe samples that can be used effectively. The one-sided sampling method targets samples that are likely to be in one of the first three groups and excludes them from the training set. This method is not parameterized, and the user has no control over the resulting class distribution.

Barandela et al. [5] developed Wilson's Editing, a resampling method that utilizes the k-nearest neighbor algorithm to guide its sampling search. Wilson's Editing classifies each sample based on the class of the three nearest neighbors. If a majority class sample is misclassified by this algorithm, it is excluded from the final data set. Though a user has no control over the final class distribution, Barandela tested two different distance metrics for comparing samples: a standard Euclidean distance algorithm, and a weighted distance algorithm. Micó et al. [60] found that using fine-tuned neighborhood classification rules other than the k-

nearest neighbor algorithm improved the performance of Wilson's Editing.

Chawla et al. [10] generated the synthetic minority over-sampling technique (SMOTE). SMOTE adds new minority sample points to the data set that are created by finding the nearest neighbors to each minority sample. The method finds some of the nearest neighbors to the current minority sample and calculates the equations for the unique lines going through each pair of the minority sample and nearest neighbor samples. Depending on the degree of over-sampling required, the method places into the data set points along some or all of these lines. These points can be placed at any point on the extrapolation lines. Chawla recommended using the first five nearest neighbors to maximize the quality of the synthetic samples. Han et al. [37] extended the SMOTE idea to only create synthetic samples for the data set that are on or near the decision boundary. Both SMOTE and Borderline SMOTE can be parameterized to oversample the minority class to virtually any degree.

Jo and Japkowicz [45] built the cluster-based over-sampling method. This method uses the k-means algorithm to cluster together similar samples. The resulting clusters have the most between-class variance and the least within-class variance possible. Those clusters that consist of only a small number of minority samples are artificially resampled. There is no control over the final class distribution, but its operation can be fine tuned by using different numbers of clusters. By using more clusters, the odds of finding some of the small disjuncts in the data set increase, but using too many clusters could lead to overfitting and too much oversampling.

2.1.3 Limits of Resampling

While many of these resampling methods can result in slightly to greatly improved performance over the original data set, there are significant issues surrounding their use. Under-sampling methods have the potential of eliminating valuable samples from consideration of the classifier entirely. Over-sampling methods, whether they duplicate existing samples or synthetically create new samples, can cause a classifier to overfit the data [13]. While many studies have shown some benefits to artificial rebalancing schemes, many classifiers are relatively insensitive to a distribution's skew [25], so the question of whether simply modifying the class ratio of a data set will always result in significant improvement is considered open by some researchers [41].

Assuming that simply adjusting a data set to a more favorable distribution can improve performance, it is difficult to determine the best distribution for any given data set. Some data sets can see strong performance with a less skewed data set, and others only see good performance from fully balanced distributions. Al-Shahib, Breitling, and Gilbert [2] used random under-sampling to 25%, 50%, 75%, and 100% of the samples required to fully balance the distribution. Even for the 75% case, the data set was skewed above a $1 : 4$ class ratio, and the performance suffered for each non-fully balanced distribution. But with 100% random under-sampling, performance increased dramatically. The fully balanced heuristic will likely give good results in comparison to the original data set, but to find the best distribution, a researcher must use an extensive model selection procedure.

Finally, there is also the question of whether resampling methods actually combat the true nature of bad class distributions. Jo and Japkowicz [45] argue that while a cursory analysis of imbalanced data sets indicates that the class distribution is the primary problem, a cluster analysis of many imbalanced data sets shows that the real problem is not truly the class imbalance. The over-arcing problem is the rare sample or small disjunct problem. The classification of small disjuncts of data is more difficult than large disjuncts because of classifier bias and the effects of feature set noise, class noise, and training set size [67]. Jo and Japkowicz's cluster-based over-sampling method [45] resulted in improved balanced accuracy of both decision tree and neural network classifiers despite not creating a fully balanced class distribution.

2.2 New Algorithms

A wide variety of new learning methods have been created specifically to combat the class imbalance problem. While these methods attack the problem in different ways, the goal of each is still to optimize the performance of the learning machine on unseen data.

2.2.1 One-class Learners

One-class learning methods aim to combat the overfitting problem that occurs with most classifiers learning from imbalanced data by approaching it from a unsupervised learning angle. A one-class learner is built to recognize samples from a given class and reject samples from other classes. These methods accomplish this goal by learning using only positive data points and no other background information. These algorithms often give a confidence level of the resemblance between unseen data points and the learned class; a classification can be made from these values by requiring a minimum threshold of similarity between a novel sample and the class [42].

One of the most prominent types of one-class learners is the one-class SVM studied by Raskutti and Kowalczyk [66]. They investigated the use of random under-sampling and different regularization parameters for each class on an SVM's generalization capability. They trained SVMs for two different tasks: similarity detection and novelty detection. Similarity detection SVMs are trained using primarily minority samples, and novelty detection SVMs are trained mainly with majority samples, but the goal for each is to identify points in the minority class successfully. On both the real-world and the synthetic data they tested, the best performance was found using the one-class SVM using only minority samples; for most of the soft margin parameters used, the difference in performance was statistically significant. They argued that the strong performance of one-class SVMs can even generalize to imbalanced, high dimensional data sets provided that the features are only weakly predictive of the class.

The other main type of one-class learner studied is the autoencoder investigated by Japkowicz [42]. The autoencoder is a neural network that is trained to reconstruct an input sample as its output. The difference between the input to the network and the output of a network is called the reconstruction error. This error is used to classify novel samples. If there is very little reconstruction error, then the sample is considered to be in the trained class; if there is a substantial amount of error, the sample is predicted to be in a different class. The results showed that autoencoders performed at least as well as supervised feed-forward neural networks on three real-world problem domains. They also argued that as long as there are enough samples in the training class, the optimal level of reconstruction error for classification can be found through model selection.

However, some research has shown poor results for one-class learners. Manevitz and Yousef [58] found that one-class SVMs had strong performances on a variety of problem domains, but the performance was not much better than other algorithms available. Additionally, they discovered that the performance of the one-class SVM was strongly affected by the choice of learning parameters and the kernel used; many of the other algorithms studied were much more stable. They recommended that further research be conducted to help researchers identify when a one-class SVM could be useful, but according to Elkan [29], no such research has been published yet.

Elkan [29] studied the use of SVMs trained using non-traditional data sets with only positive and unlabeled samples. He compared the results of an SVM trained using all of the class labels, an SVM using only positive and unlabeled data, an SVM using only postive and weighted unlabeled data, and an SVM with a soft margin parameter for each class chosen by cross-validation. His results showed that the best performance was found using all of the class labels; ignoring the labels of the unlabeled data resulted in a significant drop in true positive rate for a fixed false positive rate. He also argued that entirely discarding majority samples would lead to subpar performance compared to those using the majority samples as unlabled data because there is still information in these samples. Thus, unless one has only training samples known to be from one class and no other information, one-class learners are likely not the best approach.

2.2.2 Ensemble Methods

Ensemble methods intelligently combine the predictions of a number of classifiers and make one prediction for the class of a sample based on each of the individual classifiers. In many cases, the performance of an ensemble is much better than any of the individual classifiers in the ensemble [63]. For imbalanced data sets, where any one classifier is likely to not generalize well to the task, we may realize large improvements in performance by using many individual classifiers together. The individual classifiers in an ensemble are trained using randomly selected subsets of the full data set. As long as each subset is sufficiently different from the others, each classifier will realize a different model, and an ensemble may give a better overall view of the learning task. Research on ensemble methods has focused on two different ways to resample the data set: bagging and boosting.

Bagging was initially developed by Breiman [8]. Bagging is short for bootstrap aggregation. In a bagging ensemble, each individual classifier is trained using a different bootstrap of the data set. A bootstrap from a data set of N samples is a randomly drawn subset of N samples with replacement. The replacement allows samples to be drawn repeatedly; it can be shown that the average bootstrap will contain about 62% of the samples in the original data set. Once each of the individual classifiers is trained, the final prediction is made by taking the majority vote of the individual classifiers. Bagging works extremely well provided that the individual classifiers use an unstable algorithm that can realize large differences in the classifications from minor differences in the training data.

The most popular bagging ensemble is the random forest [9]. The random forest uses decision trees as the individual classifier. Before training the individual decision trees on their bootstraps, a random feature selection algorithm removes all but a small number of the features to increase the speed of training and the disparity between models. Because the random forest tries to maximize the accuracy of its predictions, it suffers from the class imbalance problem.

Chen, Liaw, and Breiman [14] modified the random forest algorithm in two different ways: using a modified sampling approach, and using a weighted classification scheme in both the individual trees and the final classification. The new sampling approach takes a bootstrap of the minority class data and then takes a random sample with replacement of the same size as the minority class bootstrap except from the majority class. The weighted classification scheme is a cost-sensitive method applied to the individual trees for finding split points in features and for weighting the leaves of the tree. The final classification averages the weighted votes from each tree. For a further discussion of cost-sensitive learning, see Section 2.2.3. The results of Chen's study showed that both modifications of the random forest algorithm improved performance, but there was no clear winner between the two. They recommended using the modified bootstrap approach simply because the training of the individual trees is quicker.

Boosting was introduced by Schapire [68] as a way to train a series of classifiers on the most difficult samples to predict. The first classifier in a boosting scheme is trained using a bootstrap of the data. We then test that classifier on the whole data set and determine which samples were correctly predicted and which were incorrectly predicted. The probability of an incorrect sample being drawn in the next sample is increased, and the probability of drawing a correctly predicted sample is decreased. Thus, future classifiers in the series are more likely to use samples which are difficult to classify in their training set and less likely to use those samples that are easy to classify. Schapire rigorously proved that if a classifier can achieve just above chance prediction based on the class ratio, a series of these classifiers can be used to increase the classification to any arbitrary performance level.

Chawla, Lazarevic, Hall, and Bowyer [12] created a new boosting algorithm called SMOTEBoost to better handle imbalanced data. SMOTEBoost uses a standard AdaBoost algorithm to alter the probability of trained samples, but after each sample is drawn from the data set, we apply the SMOTE resampling method to the minority class in order to improve the individual classifier's performance. They compared the SMOTEBoost algorithm to a single classifier using SMOTE, the standard AdaBoost algorithm, and to AdaBoost using SMOTE. SMOTEBoost outperformed all of the alternative algorithms by a statistically significant margin. A discussion of SMOTE can be found in Section 2.1.2.

Sun, Kamel, and Wang [69] investigated a problem similar to the class imbalance problem: the multi-class classifier problem. It is difficult for a classifier to correctly predict samples from more than two classes because the chance prediction rate goes down as the number of classes goes up. Additionally, the cost of misclassifying samples can differ depending on the predicted class of the sample. As an example, it is far worse to predict somebody with lung cancer as not having cancer than as having prostate cancer. They modified the standard AdaBoost algorithm to use a cost matrix for scoring its predictions so that those with more severe misclassification costs are the most likely to be selected. The modified AdaBoost algorithm used a genetic algorithm to determine the cost matrix before training the individual classifiers. They concluded that this algorithm outperformed the standard AdaBoost and that the offline cost of determining the cost matrix is minimal compared to the performance jump.

While the research shows that ensembles can improve performance on imbalanced data, there are some problems that users can encounter with ensembles. Bagging methods tend to improve on the performance of individual classifiers, but the improvement in performance is sometimes very small, and it is often far worse than the performance of a boosting method. However, boosting methods are not guaranteed to result in performance better than just a single classifier [63]. In fact, Schapire [68] proved that if an individual classifier cannot perform better than chance on the data set, adding more classifiers will not improve performance at all. The trivial majority classifier sets a very high mark for performance, and individual classifiers may be hard-pressed to beat the baseline accuracy of the class ratio.

2.2.3 Non-Accuracy Maximizing Algorithms

One of the primary problems with using standard machine learning algorithms to generalize about imbalanced data sets is that they were designed with global predictive accuracy in mind. Thus, regardless of whether a researcher uses the F1-measure, precision, recall, or any other evaluation statistic to measure the performance of a trained model, the algorithm will develop the trained model by attempting to maximize its accuracy.

One of the most popular classes of non-accuracy learning methods is the cost-sensitive learners. Cost-sensitive learning methods try to maximize a loss function associated with a data set. Cost-sensitive learning methods are motivated by the finding that most real-world applications do not have uniform costs for misclassifications [23]. The actual costs associated with each kind of error are unknown typically, so these methods either need to determine the cost matrix based on the data and apply that to the learning stage, or they need to take a candidate cost matrix as an input parameter. Using a cost matrix allows a learning method to be far more adaptable to the skew of a distribution.

As an example of the use of a cost matrix in a problem, consider a simple pre-cancer screening that tells a person whether they have cancer or not. There are costs and benefits attached to each result in the matrix. Diagnosing a person with cancer when they really have cancer means that they will undergo treatment, but treatment can extend the lifespan a number of years, so the cost is not very severe. Likewise, saying a person does not have cancer when they really do not costs the person only a token amount for the test. In contrast, the two errors have severe penalties. Misdiagnosing someone with cancer when they do not have cancer subjects the person to more painful tests and mental anguish, so the cost is rather high. Giving a person a clean bill of health when they really have cancer means it goes undiagnosed longer and could shorten their lifespan significantly, so this cost is the highest. A realization of the cost matrix described above is shown in Table 2.1.

Table 2.1. Cost Matrix for Cancer Prediction

Predict ↓, Actual →	Cancer	No Cancer
Cancer	5	20
No Cancer	100	1

One of the first cost-sensitive learning methods was developed by Domingos [23]. MetaCost is a wrapper method that works with an arbitrary accuracy-maximizing classifier and converts it into a cost-sensitive classifier. MetaCost works using a variant of the bagging ensemble developed by Breiman [8]. After each sub-classifier is trained on its bootstrap, it estimates the unweighted average of class probabilities from the trained models as the vote for a sample. MetaCost is designed to allow all of the trained models to influence the vote, but it can also be set to use only trained models that didn't use the sample in training. The first lowers the variance of its predictions, and the second reduces the bias. MetaCost can also be used with algorithms that don't explicitly give probabilities for their predictions.

Many of the first cost-sensitive algorithms, including the above two, are only well-understood in two-class problems. Many classification tasks allow for a large number of different classes. Abe, Zadrozny, and Langford [1] developed a method to perform cost-sensitive learning directly on multi-class data sets. This method works by iteratively searching the cost matrix space via gradient descent with data space expansion until we find the best matrix. This algorithm outperformed MetaCost and bagging in their research. A similar algorithm, called asymmetric boosting, was devloped by Masnad-Shirazi and Vasconcelos [59]. The asymmetric boosting algorithm uses gradient descent to minimize the cost-sensitive loss of the ensemble. This implementation outperformed many previous attempts at building a cost-sensitive AdaBoost algorithm on a facial detection task.

A closely related idea to cost-sensitive learners is shifting the bias of a machine to favor the minority class [27, 39, 70]. This approach assumes that even if the trivial majority classifier arises as a trained model, there are differences in the margin, or posterior probability, predicted for the sample. We can apply any cost matrix to these predictions and solve the risk function associated with this matrix to obtain the optimal decision boundaries for that matrix. Elkan [27] showed that one only needs to train one classifier and convert its predictions using a theorem and this cost matrix to a cost-sensitive classifier. This method would not choose the best cost matrix, but it would allow for more simple optimization of the cost matrix.

A new type of SVM developed by Brefeld and Scheffer [7] maximizes the AUC rather than accuracy. They empirically showed the AUC maximizing SVM improved the AUC score established by the standard SVM. However, the standard algorithm was noted to be a $O(n^4)$ algorithm and too slow for anything larger than a trivially small dataset. To make the calculations feasible, they developed an approximate solution using a k-means clustering method to group together the constraints. The approximation is a $O(n^2)$ algorithm and still improves the AUC compared to the standard SVM, but they noted that the runtime is still significantly longer than standard SVM implementations. Parallel to this work, Thorstem Joachims [46] developed an extension of his program SVM^{light}, called SVM^{perf}, that allows a user to train linear SVMs in linear time compared to the sample size. At the same time, this program can also optimize a variety of loss functions, including AUC, precision-recall break-even rate, F1-measure, and the standard accuracy metric. This makes it far more adaptable to various problem domains where other evaluation statistics are preferred to accuracy. However, neither Brefeld and Scheffer nor Joachims explicitly tested their algorithms on imbalanced data sets, so there is no guarantee that these would actually combat the class imbalance problem.

2.2.4 Limits of Algorithms

All of the above described algorithms have been shown empirically to improve on the performance of basic algorithms for some data sets. However, many of the improvements in performance, while they may be statistically significant for some α, are extremely small. In fact, Drummond and Holte [26] argue that for severely imbalanced distributions, the trivial majority classifier may be impossible to beat. This is because there is already a very high performance threshold set by the trivial majority classifier. The improvement in performance over the trivial majority classifier is called the error reduction rate. For example, if the trivial majority classifier gets 20% of its predictions wrong and a trained classifier gets only 5% of its predictions wrong, the error rate is reduced by 0.75. To get the same error reduction when the trivial majority classifier only gets 1% of its predictions wrong, a trained classifier must only get 0.25% of its predictions wrong. Large error reduction rates are difficult to achieve under the best circumstances.

The best classifier available is the Bayes optimal classifier. In cases of severe class imbalances, the Bayes optimal classifier is at least as good as the trivial majority classifier. But even the Bayes optimal classifier cannot improve results to perfection very often. Drummond and Holte [26] showed a number of results about the Bayes optimal classifier in relation to the trivial majority classifier. As the imbalance increases, the benefit to relative cost reduction decays quickly to the point where there is almost no absolute reduction in error despite a large relative reduction. The problem is even worse for the practical algorithms they studied, including nearest neighbors and decision trees. A cost-sensitive learner would alleviate the problem by changing the classification task to one with more potential for improvement, but this will inevitably lead to a large number of false alarms. An alternate solution that avoids cost-sensitive learning is redefining the definition of the minority class so that the task is more granular. This could keep the benefits of a predictor while reducing the imbalance.

2.3 Feature Selection Methods

The goal of feature selection in general is to select a subset of j features that allows a classifier to reach optimal performance, where j is a user-specified parameter. Feature selection is a key step for many machine learning algorithms, especially when the data is high dimensional. Microarray-based classification data sets often have tens of thousands of features [74], and text classification data sets using just a bag of words feature set have orders of magnitude more features than documents [32]. The curse of dimensionality tells us that if many of the features are noisy, the cost of using a classifier can be very high, and the performance may be severely hindered [13].

2.3.1 Types of Feature Selection

There are three different types of feature selection methods commonly used: metrics, wrappers, and embedded methods. Each of these types has positives and negatives that may indicate their use on a specific data set.

Feature selection metrics are used to rank features independent of their context with other features. A metric evaluates the effectiveness of each individual feature in predicting the class of each sample and then ranking the features from most helpful to least helpful to classification [36]. Koller and Sahami [49] showed that the optimal feature ranking can be calculated using Markov blankets, but this calculation is intractable. Researchers have developed a large number of rules to score features linearly and maximize the speed of their calculations. This speed is the metric's biggest strength, but because they do not look for interactions between features, they can only select a set of strong features and not an optimal feature set.

Wrappers choose a feature subset that best predicts the outcome based on how these features interact with each other. They accomplish this by using a learning method to evaluate the predictive performance of different feature subsets chosen by a searching heuristic. Because a wrapper treats the learning machine as a black box, any inductive learner may be used with a wrapper. Unstable learning methods, like neural networks and decision trees, tend to see the most improvement, though any learning method can see improved results [48]. There are a number of problems with using wrappers. Even the most efficient of search heuristics must develop a trained model for each candidate feature set, and the number of potential feature subsets increases exponentially with the number of features. Another problem was researched by Loughrey and Cunningham [53]; if a wrapper is used naïvely until the best feature set cannot be improved on, the selected feature set has a tendency to overfit the data severely. Finally, to perform feature selection with a wrapper, the data set must be partitioned into training, validation, and testing subsets. For a data set that already has a small sample size, this makes learning even more difficult.

Embedded methods are fairly similar to wrappers in that they choose a subset of features that best predict the outcome. The difference is that while wrappers go around the learning machine and must retrain the method for each potential subset, embedded methods work in conjunction with the learning machine to choose the best feature subset. This helps avoid multiple iterations of retraining, but it also means that the learning method must be written with the feature selection method in mind. This restriction severely limits the number of embedded methods available [36]. The most popular embedded method is the recursive feature elimination algorithm built into the SVM [20].

Because the class imbalance problem is commonly accompanied by the issue of high dimensionality of the data set [13], applying feature selection techniques is a necessary course of action. Ingenious sampling techniques and algorithmic methods may not be enough to combat high dimensional class imbalance problems.

Van der Putten and van Someren [22] analyzed the data sets from the CoIL Challenge 2000 and found that feature selection was more significant to strong performance than the choice of classification algorithm and helped combat the overfitting problem the most. Forman [32] noted a similar observation on highly imbalanced text classification problems and stated ”no degree of clever induction can make up for a lack of predictive signal in the input space.”

However, another analysis of the CoIL Challenge 2000 by Elkan [28] found that feature selection metrics were not good enough for this task; instead, the interaction between different features also needed to be considered in the selection phase. The biggest flaw that he found with most of the applied feature selection methods is that they did not consider selecting highly correlated features because they were thought to be redundant. Guyon [36] gave a strong theoretical analysis as to the limits of feature selection metrics. She showed that irrelevant features on their own can be useful in conjunction with other features, and the combination of two highly correlated features can be better than either feature independently.

The best feature selection methods should clearly consider the interactions between the selected features. However, this kind of feature selection method forces us to consider all possible subsets, and such a method has an exponential run-time and is intractable. Thus, we cannot use wrappers or embedded methods. Instead, we must use feature selection metrics with high dimensional data sets. Feature selection metrics have a linear run-time because they evaluate each feature once and then select the best ranking features. Additionally, they also have the benefit of avoiding overfitting to the data set; while a selected feature set may increase the bias marginally, the decrease in variance is significantly larger, so the error will usually drop [36].

2.3.2 RELIEF

One of the most popular and well-researched feature selection metrics is an algorithm designed by Kira and Rendell called RELIEF [47]. RELIEF evaluates features using the nearest neighbor rule based on how well a feature's values differentiate themselves from nearby points. When RELIEF selects any specific instance, it searches for two nearest neighbors: one from the same class (the nearest hit), and one from the other class (the nearest miss). Then RELIEF calculates the relevance of each attribute A by the rule found in Equation (2.1). The justification for this rule is that if a feature is a good predictor, instances from different classes should have vastly different values, and instances of the same class should have very similar values. Unfortunately, the true probabilities cannot be calculated, so we must estimate the difference in Equation (2.1). This is done by calculating the distance between random instances and their nearest hits and misses. For discrete variables, the distance is 0 if the same and 1 if different; for continuous variables, we use the standard Euclidean distance. We may select any number of instances up to the number in the set. The more instances we select, the better the approximation [50]; Common implementations today search the entire sample set to get the best approximation possible. Algorithm 1 details the pseudo-code for implementing RELIEF.

$$\begin{aligned} W(A) &= P(\text{different value of A}|\text{nearest miss}) \\ &\quad -P(\text{different value of A}|\text{nearest hit}) \end{aligned} \tag{2.1}$$

RELIEF has been extended by researchers to make it useful in a wide variety of applications. Kononenko [50] implemented checking multiple nearest hits and misses, handling missing values in three different manners, and the capacity to

Algorithm 1 RELIEF

```
Set all $W(A) = 0$
for $i = 1$ to $m$ do
    Select instance $R$ randomly
    Find nearest hit $H$ and nearest miss $M$
    for $A = 1$ to $numfeats$ do
        $W(A) = W(A) - dist(A, R, H)/m$
        $W(A) = W(A) + dist(A, R, M)/m$
    end for
end for
```

handle multi-class data. Sun [69] re-implemented RELIEF as an iterative algorithm similar to the EM algorithm that simultaneously checks multiple nearest hits and misses and corrects the objective function so that it can handle outlier data points. Other researchers have used RELIEF in multi-stage feature selection algorithms because it removes irrelevant features very well; the other steps handle redundancy and interaction between the features [4, 6].

Despite the popular use of RELIEF in the machine learning community, it does not successfully handle imbalanced data very often. This was shown by Chen and Wasikowski [18] where RELIEF performed significantly worse than both the other feature selection metrics studied and the baseline linear SVM using the entire feature set. RELIEF did perform better than the baseline nearest neighbor algorithm, but it performed worse than their new algorithm. RELIEF cannot handle small classes very well because as the class size shrinks, there is a greater chance that a sample will have an outlier feature value that throws off the relevance approximation. RELIEF measures a feature's relevance through a local view only considering feature values near the current value. In imbalanced data, it is imperative to examine all of the values for a feature together to make a more global prediction of its goodness.

2.3.3 Feature Selection on Imbalanced Data

Feature selection as a general part of machine learning and data mining algorithms has been thoroughly researched [15,17,35,36,65,73,75], but its importance to resolving the class imbalance problem is a recent development with most research appearing in the previous seven years. In this time period, a number of researchers have conducted research on using feature selection to combat the class imbalance problem [32, 78].

Mladenić and Grobelnik [62] studied the performance of feature selection metrics in classifying text data drawn from the Yahoo web hierarchy. They applied nine different metrics to the data set and measured the power of the best features using the naïve Bayes classifier. Their results showed that the odds ratio (OR) was nearly universally the best performing metric on the F-measure with $\beta = 1$ and 2, precision, and recall measures; in general, they concluded that the best metrics choose common features and consider the domain and learning method's characteristics. This explains why OR would be very good for the naïve Bayes classifier: it tries to maximize the posterior probability of a sample being in a class.

Forman [32] examined the classification power of feature selection metrics on a number of text sets drawn from the TREC competitions, MEDLINE, OHSUMED, and Reuters databases. He used the features selected by different metrics to train linear SVMs and evaluated their performance using accuracy, precision, recall, and the F1-measure. While a small number of metrics saw mild improvements over the performance of an SVM using all of the available features, the best overall performance came from features selected by bi-normal separation (BNS). Forman's general finding is that the metrics that performed best selected features that separated the minority class from the majority class well. He also concluded that the evenly balanced form of BNS that selected features with equal weight to true positive rate and false positive rate gave the best performance.

Zheng, Wu, and Srihari [78] analyzed how the types of selected features affected classification performance. Positive features are those that indicate membership in a class, and negative features are those that indicate lack of membership in a class. They used the chi-square statistic (CHI), information gain (IG), and OR algorithms as a starting point to create one-sided and two-sided metrics; one-sided metrics only select positive features on their score, and two-sided metrics select both positive and negative features based on the absolute value of their score. They then compared the performance of the one and two-sided metrics with different ratios of the best positive and negative features. The ratio selection method resulted in improved performance on the break-even point between precision and recall compared to general one and two-sided metrics. Thus, both positive and negative features are important to obtain the best possible power from a classifier.

2.3.4 Issues with Feature Selection

One common problem with many of the standard feature selection metrics used in the previous studies [32, 62, 78] is that they operate on Boolean data. Some metrics, such as IG and CHI, generalize well to nominal data, but they break down when handling continuous data. Thus, when one uses a discrete or Boolean feature selection metric on continuous data, its performance is entirely dependent on the choice of the preset threshold used to binarize the data. This threshold determines the confusion matrix's true positive (TP), false negative (FN), false positive (FP), and true negative (TN) counts.

Consider a classification problem using two different feature sets such that a classifier using the first feature set yields a higher TP count and lower TN count than the second set. However, if we bias the classifier to change its decision threshold, that classifier may instead have a higher TP count and lower TN count for the second feature set. Thus, it is impossible to tell which feature set is better because one threshold will not give us adequate information about the classifier's performance. This happens because the confusion matrix is an evaluation statistic of a classifier's predictive power based solely on this one threshold [56]. This can be seen if we plot the receiver operating characteristic (ROC) associated with each feature set. One ROC is said to dominate another if the second ROC is less than or equal to the first for every point [21]; a dominating ROC is better than the other in all cases, and we would prefer the use of the dominating classifier. In our scenario, neither ROC would dominate the other, so we cant tell which is better.

It may be advantageous to use a feature selection metric that is non-parametric and could use all possible confusion matrices when using ordinal or ratio data. Thus, it would be possible to find the threshold that results in the highest performance possible. One possibility is the ROC. This is a non-parametric measure of a system's power that compares the true positive rate with the false positive rate. We can then quantify the ROC using the AUC and apply this score as our feature ranking. The FAST metric [18] evaluates features using an approximation to the AUC. Both linear SVM and nearest neighbor classifiers using FAST-selected features saw improved AUC and balanced error rates over the baseline score using all features and the scores achieved using RELIEF and Pearson correlation coefficient (PCC)-selected features.

Another non-parametric statistic for evaluating the performance of a classifier is the precision-recall (P-R) curve. Davis and Goadrich [21] noted that the ROC can sometimes give an overly optimistic idea of the performance of a learning method when the data set is extremely imbalanced. A classifier that has a very strong ROC and may appear to be close to perfect can have a rather weak P-R curve that shows much room for improvement. For highly imbalanced data, it may be advantageous to modify the FAST metric to use the area under the P-R curve (PRC) as the evaluation statistic rather than AUC.

Chapter 3

Method

Much of this report examines the performance of various feature selection metrics and how they combat the class imbalance problem. Because there are two other general approaches to handling this problem, we must also examine the performance of resampling methods and new algorithms.

Features are classified as either positive or negative. Positive features are those that, when present, indicate membership in a class, while negative features indicate non-membership in a class [78]. In the context of continuous data, a larger positive feature value indicates membership in a class, and a larger negative feature value indicates non-membership in a class. As a part of his study on feature selection for text classification, Forman [32] investigated how a preference for either more positive or negative features affected the precision and recall of his learning methods. He concluded that positive features influence the precision of a classifier, and negative features improve the recall of a classifier. The best F1-measure performance was found by balancing the positive and negative features completely.

Feature selection metrics are either one-sided or two-sided. They differ based

Table 3.1. Feature Selection Formulas

Name	Formula
CHI	$t(tp,(tp+fp)P_{pos})+t(fn,(tn+fn)P_{pos})$ $+t(fp,(tp+fp)P_{neg})+t(tn,(tn+fn)P_{neg})$
IG	$e(pos,neg)-((\frac{tp+fp}{N})e(tp,fp)+(\frac{tn+fn}{N})e(fn,tn))$
OR	$\log\frac{tptn}{fpfn}$
PCC	$\frac{1}{N-1}\Sigma(\frac{X-\mu_X}{\sigma_X})(\frac{Y-\mu_Y}{\sigma_Y})$
S2N	$\frac{\mu_1-\mu_{-1}}{\sigma_1+\sigma_{-1}}$
FAST	Area Under ROC
FAIR	Area Under PRC

CHI, IG, OR binarize the data using $\theta = 1/2(\mu_1+\mu_{-1})$ as cutpoint
$t(c,e)=\frac{(c-e)^2}{e}$, $P_{pos}=\frac{pos}{N}$, $P_{neg}=\frac{neg}{N}$, e is conditional entropy

on whether they select just positive features or a combination of positive and negative features. A one-sided metric uses the signed value of a feature's score, and will thus only select positive features. A two-sided metric ignores the sign of a feature's score and only considers the absolute value; it can select both positive and negative features [78]. Forman [32] and Zheng [78] both agree that, in general, two-sided metrics are better than one-sided metrics because the negative features improve the recall, or true positive rate, of the classifier; the minority class is the most important class to get correct predictions because of the class distribution. However, there are occasionally combinations of data sets and metrics that result in the one-sided metrics having superior performance.

3.1 Binary Feature Selection Metrics

The following feature selection metrics can handle binary data; some of these metrics can also handle nominal data, but these were forced to use binary data to ensure no metric had an advantage because of a feature's structure and how it was discretized. Because the data sets we studied consist of continuous or discrete data, we must preprocess the data before applying these metrics. We used the binzarization method described by Guyon and Elisseeff: after finding the mean feature value for the two classes, we set a threshold θ at the midpoint between the two mean values. The features are then binarized according to this threshold [36]. We use the $\geq$ relation to resolve ties with θ.

3.1.1 Chi-Square Statistic

CHI is a statistical test. For the purposes of feature selection, CHI measures the independence of a feature from the class labels based on the confusion matrix. It is a two-sided metric. CHI generalizes well for nominal data, but it does not work on continuous data. This happens because there is zero probability for an exact value to be drawn from a continuous distribution. Even assuming that there is a non-zero probability for an exact value to be drawn from the distribution, in practical cases, we will typically have one sample for each feature value, and this leads to extremely small expected counts of feature values. Forman noted that this test can behave erratically when there are small expected counts of features; this is fairly common with imbalanced data sets [32], and it is also common in small samples. The formula for calculating CHI can be found in Equation (3.1).

$$\begin{aligned} \text{CHI} &= t(tp,(tp+fp)P_{pos}) + t(fn,(tn+fn)P_{pos}) \\ &\quad + t(fp,(tp+fp)P_{neg}) + t(tn,(tn+fn)P_{neg}) \\ t(c,e) &= \tfrac{(c-e)^2}{e} \end{aligned} \tag{3.1}$$

3.1.2 Information Gain

IG is the name for the Kullback-Leibler divergence in machine learning. As a feature selection metric, IG measures the decrease in entropy of the class labels when we use a feature. The entropy of a random variable, such as the class labels, measures how uncertain we are about the values drawn from a random variable. As the proportion of samples approaches fully balanced, the entropy grows. The conditional entropy measures the remaining uncertainty for a random variable when we know the values of another random variable, such as the features in a data set. Once the entropy and conditional entropy are calculated, we simply subtract the two. This measure is two-sided. Like CHI, it generalizes for nominal data but cannot handle continuous data for similar reasons. The formula for calculating IG can be found in Equation (3.2).

$$\begin{aligned} \mathrm{IG} &= e(pos, neg) - ((\tfrac{tp+fp}{N})e(tp, fp) + (\tfrac{tn+fn}{N})e(fn, tn)) \\ e(x, y) &= -\tfrac{x}{x+y}\log\tfrac{x}{x+y} - \tfrac{y}{x+y}\log\tfrac{y}{x+y} \end{aligned} \tag{3.2}$$

3.1.3 Odds Ratio

OR is a descriptive test that analyzes the occurrence of an event given that another event happened already. In machine learning, we use it to quantify the change in odds of a sample being drawn from a class given a feature's values. To calculate this change, we find the odds of a feature occuring in the positive class and normalize by the odds of the feature occuring in the negative class. OR can be made into a one-sided or a two-sided metric very easily. If we have a zero count in the denominator, we replace the denominator with 1. This is consistent with how Forman computed his two-sided OR [32]. In a pilot study, we compared the two-sided and one-sided OR algorithms on our biological data sets. The two-sided modification of the algorithm takes the log of the ratios and then squares these log values. The pilot study found that the one-sided OR performed better on our data sets across all feature counts. Thus, we used the one-sided OR rather than the two-sided. This metric is designed to operate solely on binary data sets. The formula for calculating OR can be found in Equation (3.3).

$$\mathrm{OR} = \log \frac{tp\ tn}{fp\ fn} \tag{3.3}$$

3.2 Continuous Feature Selection Metrics

The following feature selection metrics are designed to operate on continuous data and do not require any preprocessing of the data.

3.2.1 Pearson Correlation Coefficient

PCC is a statistical test that measures the strength and quality of the relationship between two variables. Correlation coefficients can range from -1 to 1. The absolute value of the coefficient gives the strength of the relationship; absolute values closer to 1 indicate a stronger relationship. The sign of the coefficient gives the direction of the relationship. If it is positive, then the two variables increase or decrease with each other. When it is negative, one variable increases as the other decreases.

In machine learning problems, PCC is used to evaluate how accurately a feature predicts the target independent of the context of other features. The features are then ranked based on the correlation score [36]. For problems where the covariance ($cov(X_i, Y)$) between a feature (X_i) and the target (Y) and the variances of the feature ($var(X_i)$) and target ($var(Y)$) are known, the correlation can be directly calculated; this can only be used when the true values for the covariance and variances are known. When these values are unknown, an estimate of the correlation can be made using Pearson's product-moment correlation coefficient over a sample of the population. The Pearson correlation coefficient is a one-sided metric, but a two-sided metric can be created by taking the square of each feature's score. This formula only requires finding the mean of each feature and the target to calculate. The formula for calculating PCC can be found in Equation (3.4).

$$\mathrm{PCC} = \frac{\Sigma_{k=1}^{m}(x_{k,i} - \overline{x_i})(y_k - \overline{y})}{\sqrt{\Sigma_{k=1}^{m}(x_{k,i} - \overline{x_i})^2 \Sigma_{k=1}^{m}(y_k - \overline{y})^2}} \tag{3.4}$$

3.2.2 Signal-to-noise Correlation Coefficient

The signal-to-noise ratio is originally a concept in electrical engineering. It is defined as the ratio of a signal's power to the power of the noise present in the signal. If a signal has a lot of noise present, it is much more difficult to isolate the signal. The signal-to-noise correlation coefficient (S2N) is a similar measurement. It compares the ratio of the difference between the class means to the sum of the standard deviations for each class. If, for a given feature, the two class means are distant from each other, there is less chance of a sample being drawn from the other class. If the class means are close, there's a high chance of mislabeling. Larger or smaller standard deviations scale the distance appropriately. Very little research has used this correlation coefficient as a feature selection metric; it was applied to leukemia classification with strong results [34]. This is a one-sided metric. The equation for calculating S2N can be found in Equation (3.5).

$$S2N = \frac{\mu_1 - \mu_{-1}}{\sigma_1 + \sigma_{-1}} \tag{3.5}$$

3.2.3 Feature Assessment by Sliding Thresholds

Most single feature classifiers set the decision boundary θ at the mid-point between the mean of the two classes [36]. This may not be the best choice for the decision boundary. By sliding the decision boundary, we can increase the number of true positives we find at the expense of classifying more false positives. Alternately, we could slide the threshold to decrease the number of true positives found in order to avoid misclassifying negatives. Thus, no single choice for the decision boundary may be ideal for quantifying the separation between two classes.

We can avoid this problem by classifying the samples on multiple thresholds and gathering statistics about the performance at each boundary. If we calculate the true positive rate and false positive rate at each threshold, we can build an ROC and calculate the AUC. Because the AUC is a strong predictor of performance, especially for imbalanced data classification problems, we can use this score as our feature ranking: we choose those features with the highest AUCs because they have the best predictive power for the dataset.

By using a ROC as the means to rank features, we have introduced another problem: how do we calculate the AUC? The standard method of finding the ROC involves using every possible threshold and calculating the true positive rate and false positive rate for these thresholds. In the worst case, where every feature value is different, we would have $N + 1$ thresholds on this curve, where N is the number of samples in the dataset: two would be equivalent to trivially classifying everything as one of the two classes, and $N - 1$ thresholds would go between each consecutive pair of values. For datasets with even a relatively large number of samples, this calculation is infeasible for large feature sets. If we instead approximate the ROC using a parameter $K \ll N$, for the number of thresholds, we can speed up this process significantly. Chen and Wasikowski [18] found that the approximate AUC scores using $K = 10$ thresholds were within ±0.02 of the exact AUC score for over 99% of the features and within ±0.005 for over 50% of the features for a number of microarray and mass spectrometry data sets. Thus, there is very little information lost from using the approximate method, and we save a significant amount of calculation time as well.

Once the number of thresholds is selected, we need to decide where to place the thresholds. If there are a large number of samples clustered together in one region, we would like to place more thresholds between these points to find how separated the two classes are in this cluster. Likewise, if there is a region where samples are sparse and spread out, we want to avoid placing multiple thresholds between these points so as to avoid placing redundant thresholds between two points. One possible solution is to use a histogram to determine where to place the thresholds. A histogram fixes the bin width and varies the number of points in each bin. This method does not accomplish the goals detailed above. It may be the case that a particular histogram has multiple neighboring bins that have very few points. We would prefer that these bins be joined together so that the points would be placed into the same bin. Likewise, a histogram may also have a bin that has a significant proportion of the points. We would rather have this bin be split into multiple different bins so that we could better differentiate inside this cluster of points.

We use a modified histogram, or an even-bin distribution, to correct both of these problems. Instead of fixing the bin width and varying the number of points in each bin, we fix the number of points to fall in each bin and vary the bin width. This even-bin distribution accomplishes both of the above goals: areas in the feature space that have fewer samples will be covered by wider bins, and areas that have many samples will be covered by narrower bins. We then take the mean of each sample in each bin as our threshold and classify each sample according to this threshold. Algorithm 2 details the pseudocode for implementing FAST.

Davis and Goadrich [21] argue that for extremely imbalanced data sets, the ROC can give a researcher an overly optimistic view of a classifier's performance. They show an example where two different algorithms have nearly identical ROCs for the same data set with almost identical AUC's, but the PRCs are vastly different and strongly indicate the use of one algorithm over the other. We examined the use of the P-R curve instead of the ROC as our non-parametric statistic. This modifcation is called Feature Assessment by Information Retrieval (FAIR) because it uses the information retrival standard evaluation statistics of precision and recall to build the curve. The algorithm is otherwise the same as FAST.

FAST and FAIR are both two-sided metrics. This is accomplished by examining the ROC and P-R curves built by starting from each direction and taking the maximum of the two areas. This calculation is trivial for the ROC; we simply need to take the maximum of the area calculated and 1 minus the area calculated. For the P-R curve, we simply take a parallel tabulation of the precision and recall for the majority class, build the P-R curve from these values, and take the maximum area.

3.3 SMOTE and Random Under-Sampling

Section 2.1 detailed a wide variety of different sampling techniques that aim to balance the data and make it more likely that a classifier will make good predictions. Many researchers have shown the effectiveness of simple random

Algorithm 2 FAST

```
$K$: number of bins
$N$: number of samples
$M$: number of features
$Split = 0$ to $N$ with a step size $N/K$
for $i = 1$ to $M$ do
    $X$ = Vector of samples' values for feature $j$
    Sort $X$
    for $j = 1$ to $K$ do
        $Bottom = round(Split(j)) + 1$
        $top = round(Split(j + 1))$
        $M = mean(X(bottom \text{ to } top))$
        Classify $X$ using $M$ as threshold
        $tpr(i, j) = tp/\#positive$
        $fpr(i, j) = fp/\#negative$
    end for
end for
Calculate area under ROC by $tpr$, $fpr$
```

under-sampling to fully balance the class distribution [2,16,51]. In fact, a survey of different sampling techniques by Van Hulse, Khoshgoftaar, and Napolitano [40] showed that random under-sampling is often the best sampling technique even when the imbalance is very severe. However, the data sets we are examining are extremely small, and full random under-sampling with a small imbalanced data set can result in overfitting of the classifier. To make this comparison a little more fair, we use a method studied by Chawla that combines SMOTE and full random under-sampling [10]. By doing this, we double the size of the minority class so that we lose less of the minority class information when we under-sample. The under-sampling is performed on the larger of the two classes after SMOTE has added samples so that we always end up with a fully balanced distribution.

3.4 AUC Maximizing SVM

Section 2.2 covered many of the new algorithms constructed specifically to combat the class imbalance problem. The one-class SVM is a popular approach [66], but it suffers from a problem of discarding all information about the negative class. As stated by Manevitz and Yousef [58] and later by Elkan [29], if we have samples in a class, it's harmful to classifier performance to not use them as part of the learning process. Cost-sensitive learners are strong algorithms, but Elkan [27] has shown that we can find a cost-sensitive learner by adjusting a standard classifier using rules to optimize the decisions from this classifer. Bagging methods, on average, do not improve on the baseline performance by a significant margin very often. Finally, boosting algorithms require that we have an algorithm that has better than chance prediction on its own, but we've shown that many algorithms cannot beat the trivial majority classifier on their own. The only algorithm discussed that hasn't been evaluated on imbalanced data before is the AUC maximizing SVM, and it has been shown in research to optimize AUC better than the standard SVM. Despite not being explicitly tested on imbalanced data before, we use the AUC maximizing SVM as our example algorithm.

Chapter 4

Experimental Procedure

The experimental procedures are provided in this section. We discuss our choices of induction methods, evaluation statistics, data sets, and questions we aim to answer in this paper.

4.1 Induction Methods

The overall goal of this paper is to show the effectiveness of feature selection in combating the class imbalance problem. In order to evaluate a feature selection method, we must evaluate the performance of a classifier using the feature set selected by this method and compare it to the performance of the same classifier without using feature selection. There are a lot of classifiers commonly used in machine learning, and classifiers perform differently with the exact same feature set. Thus, to measure the quality of a feature selection metric, it is not sufficient to simply select one classifier. We must evaluate the feature set on different classifiers with different biases to truly measure the strength of a feature selection method.

Previous research on feature selection for imbalanced data has used a number of different classifiers to varying degrees of performance. The linear SVM is a strong and stable algorithm, and these qualities make it fairly resistant to feature selection. Forman [32] used the SVM in his study on text classification and found moderate improvement in performance with some feature selection methods. Conversely, the naïve Bayes classifier and the nearest neighbor algorithm are weaker algorithms in general, and their performance can very greatly with even minor changes to the training data used. These traits make them prime candidates for improvement with feature selection; Mladenić and Grobelnik [62] found significant improvements in performance with the naïve Bayes classifier after using OR-selected features, and Chen and Wasikowski [18] used the nearest neighbor algorithm with RELIEF-selected features on imbalanced data successfully.

4.1.1 Support Vector Machine

One of the simplest ways to classify points into two class is to assume that the classes can be linearly separated from each other. By using this assumption, we can find a discriminant between the samples in each class without knowing anything about the distribution of training samples. A linear discriminant for classifying follows the formula $g(\mathbf{x}|\mathbf{w}, w_0) = \mathbf{w}^T\mathbf{x} + w_0$. To learn a linear discriminant, we only need to learn the parameters $\mathbf{w}$ and w_0, or the weight vector and the bias [3].

One problem with learning linear discriminants is that there are potentially many different weight vector and bias combinations that could correctly classify the training data. If each of these is correct, then we need to make an additional assumption to select one of these discriminants. The SVM is a linear discriminant classifier that adds the assumption that the best discriminant maximizes the distance from the separating hyperplane formed by the discriminant to samples on both sides. Such a hyperplane, if it exists, is called the optimal separating hyperplane or the maximum-margin hyperplane.

We start with a set of data $\mathcal{X} = \{(\mathbf{x}_i, c_i)\}$, where each $\mathbf{x}_i$ is a training sample and c_i is set to ± 1 for the class of the associated sample. We can write the hyperplane as $\mathbf{w}^T\mathbf{x} + w_0 = 0$. Remember that the goal is to select the weight vector and bias that maximally separate the data; these two parallel hyperplanes with the maximum margin between them can be expressed as $\mathbf{w}^T\mathbf{x} + w_0 = \pm 1$. If we account for the class for each sample, we can rewrite this as $c_i(\mathbf{w}^T\mathbf{x} + w_0) = 1$. For each sample, we would like to see each point be at least as distant as the margin. Thus, we must satisfy the equation $\forall i \ c_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1$. This is an optimization task where we minimize $||\mathbf{w}||$ subject to $\forall i \ c_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1$. This optimization task is difficult to solve because it depends on $||\mathbf{w}||$ which involves a square root. If we alter the equation to use $\frac{1}{2}||\mathbf{w}||^2$ instead, the solution stays the same. This optimization task can be solved by standard quadratic programming optimizers [3].

If we rewrite this optimization problem in its dual form, we observe that this task is only dependent on those samples that lie exactly on the margin. These samples are the support vectors. The dual form optimization task is to maximize the formula found in Equation (4.1).

$$\begin{aligned} &\max\ -\tfrac{1}{2}\Sigma_{i,j}\alpha_i\alpha_j c_i c_j \mathbf{x}_i^T\mathbf{x}_j + \Sigma_i\alpha_i \\ &\text{subject to } \alpha_i \geq 0 \text{ and } \Sigma_i\alpha_i c_i = 0 \end{aligned} \tag{4.1}$$

There is a significant problem with the optimization task in Equation (4.1): it cannot handle any incorrectly classified samples. If there is no perfect linear discriminant, the above formulation will fail. Cortes and Vapnik [19] modified the SVM algorithm to allow for samples to be incorrectly classified using a soft margin. This method classifies as many samples correctly as is possible and maximizes the margin for the correctly classified samples. To use the soft margin, they introduced slack variables, noted as ξ_i, to measure the deviation from the margin. These slack variables allow for some number of samples to be misclassified or to be within the margin. The dual function requires one additional parameter, noted as C, to balance between the number of support vectors and the number of non-separable points. The problem can then be stated as in Equation (4.2).

$$\begin{aligned} &\max \ -\tfrac{1}{2}\Sigma_{i,j}\alpha_i\alpha_j c_i c_j \mathbf{x}_i^T\mathbf{x}_j + \Sigma_i\alpha_i \\ &\text{subject to } 0 \leq \alpha_i \leq C \text{ and } \Sigma_i\alpha_i c_i = 0 \end{aligned} \tag{4.2}$$

The standard SVM is a linear classifier, but if we replace the dot product $\mathbf{x}_i^T\mathbf{x}_j$ with any non-linear kernel function, we can also train the SVM to find the maximum-margin hyperplane in a non-linear feature space. This allows the trained hyperplane in the original feature space to be a non-linear plane; this can improve performance in many problems. The most popular kernels include polynomials, radial basis functions, and hyperbolic tangents [3]. In our research, we only used the linear SVM.

4.1.2 Naïve Bayes Classifier

Instead of finding a discriminant and using that as a classifier, we can instead develop a probability model using the features as conditions for the probability of a sample being drawn from a class. In a probability model, we would like to find $p(C|F_1,\dots,F_n)$, where each F_i is the value for each feature and C is the class of the sample. This is commonly called the posterior. Once we have the posterior for each class, we assign a sample to the class with the highest posterior. It is difficult, if not impossible, to find the posterior directly. However, if we use Bayes' rule, we can express the posterior as a ratio of the prior times the likelihood over the evidence. Formally, this is expressed as in Equation (4.3).

$$p(C|F_1,\ldots,F_n) = \frac{p(C)p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)} \tag{4.3}$$

Practically, the evidence is not important as it does not depend on the class and, for a given sample, it remains constant. Thus, we only need to determine the prior and likelihoods to find the posterior. The prior probability is simple to calculate: we just use the training data set's class distribution. Determining the likelihood is much more difficult because the conditional probability is dependent on all of the features. If we naïvely assume that all of the features are conditionally independent of every other feature, then we simplify the likelihood formula to $p(F_1,\ldots,F_n|C) = \Pi_i p(F_i|C)$. Each of the $p(F_i|C)$ are much easier to handle, and if we assume that the features follow a given distribution, the parameters for these conditional probabilities can be easily estimated using maximum likelihood estimation (MLE) [3].

It is surprising that the simplifying assumption of complete conditional independence of the features does not degrade the performance of the naïve Bayes classifier much. Zhang [76] explained the strong classification of this algorithm by examining the dependencies between features. Consider a two-feature classification problem where each of the features depends on the other. If the dependence between these features is equally distributed among the two classes, then even though there is no conditional independence between the features, naïve Bayes is still the optimal classifier. They also looked at the distribution of dependencies in higher dimensional data sets and found that when using a large number of features together, even though most of the pairs of features were not conditionally independent, each of the dependencies č̈ancel each other outš̈o that they don't affect the classification as much. Thus, even when there are strong dependencies between pairs of features, the naïve Bayes classifier will still perform fairly well.

4.1.3 Nearest Neighbor

The nearest neighbor algorithm is an instance-based and lazy learning algorithm. Instance-based learning algorithms build their hypotheses about the classes of samples directly from the training data instead of calculating statistics and using these statistics to make hypotheses. Lazy learning algorithms defer the computation for classifying a sample until a test sample is ready to be classified. The nearest neighbor algorithm meets these criteria by storing the entire training set in memory and calculating the distance from a test sample to every training sample at classification time; the predicted class of the test sample is the class of the closest training sample [61].

The nearest neighbor algorithm is a specific instance of the k-nearest neighbor algorithm where $k = 1$. In the k-nearest neighbor algorithm, when we get a test sample we would like to classify, we tabulate the classes for each of the k closest training samples and predict the class of the test sample as the mode of the training samples' classes. The mode is the most common element of a set. In binary classification tasks, k is normally chosen to be an odd number in order to avoid ties. Selecting the best value of k is difficult, and it is even more problematic when dealing with imbalanced data. The imbalance between the two classes makes it likely that more of the k nearest training samples will be found in the majority class as k increases. We used $k = 1$ because this value is the most fair to the minority class.

Nearest neighbor algorithms can use any metric to calculate the distance from a test sample to the training samples. A metric is a two-argument function $d(x, y)$ that is non-negative, reflexive, symmetric, and satisfies the triangle inequality. The standard metric used in nearest neighbor algorithms is Euclidean distance. The main weakness of using Euclidean distance is that the variance in different features is not accounted for; the Mahalanobis distance is a scale-invariant version of the Euclidean distance that accounts for each feature's variance [57]. The formula for Euclidean distance is found in Equation (4.4), and the formula for Mahalanobis distance is found in Equation (4.5). In the Mahalanobis distance formula, S refers to the sample covariance matrix.

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4.4}$$

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T S^{-1} (\mathbf{x} - \mathbf{y})} \tag{4.5}$$

One issue with standard implementations of the nearest neighbor algorithm is that they usually only tell a user what the nearest sample's class is, but they do not give any information about its proximity to that sample or to samples from the other class. This is a consequence of the nearest neighbor algorithm being a local predictor. Distant samples have no effect on the classification of a sample. For binary evaluation statistics, this is not a problem, but because the SVM and naïve Bayes classifier both give users margins for their predictions, we would like to get a margin from this algorithm as well. We modified the implementation to give us a margin by finding the closest training sample in each of the minority and majority classes. The margin is defined as the difference between the two distances. For a given training sample, if the nearest minority sample is not much more distant than the nearest majority sample, we would be much less confident of the minority prediction than if the nearest majority sample is very distant. This margin calculation still gives identical predictions, and it also allows us to use margin-based evaluation statistics.

4.2 Evaluation Statistics

On extremely imbalanced data sets, algorithms will be hard pressed to classify test samples as members of the minority class because the discriminant scores given by the classifier are often weighted toward the majority class. Accuracy is clearly a poor measure of the performance of a classifier on imbalanced data thanks to the trivial majority classifier. There are a number of statistics that researchers commonly use to focus on the minority class, including precision, recall, and the F-measures.

Precision is the ratio of true positive predictions to all positive predictions. It measures how well a classifier performs at making positive predictions. A classifier with high precision would have a large number of true positive predictions and a small number of false positive predictions. The formula for calculating the precision of a classifier can be found in Equation (4.6), where tp refers to true positives and fp refers to false positives.

$$\mathrm{P} = \frac{tp}{tp + fp} \tag{4.6}$$

Recall, or sensitivity, is the ratio of true positive predictions to all positive samples. It measures how well a classifier performs at actually identifying a positive sample as positive. A classifier with high recall would have a large number of true positive predictions and a small number of false negative predictions. The formula for calculating the recall of a classifier can be found in Equation (4.7), where tp refers to true positives and fn refers to false negatives.

$$\mathrm{R} = \frac{tp}{tp + fn} \tag{4.7}$$

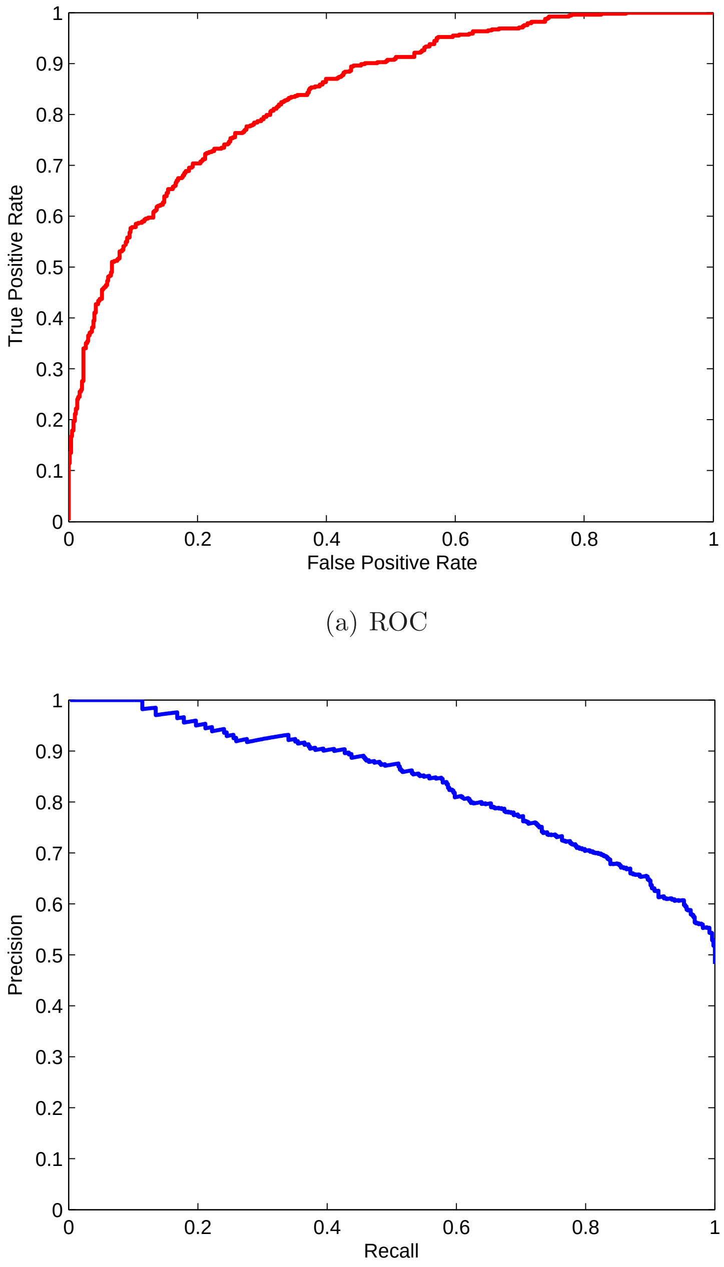
The F-measure is a family of scores that use both precision and recall to evaluate the performance of a clasifier. The F-measure is parameterized by a value, called β, that weights the effect of precision and recall on the score. The value of β must be positive; as β trends towards 0, precision gets weighted more than recall, and recall gets weighted more than precision as β goes to inf. Most researchers use $\beta = 1$ which sets precision and recall to equal weight. Other values of β may be better to use in some cases: precision may be favored in search engine results and spam filtering, while recall could be more important in cancer diagnosis. The general F-measure is calculated as in Equation (4.8), and the F1-measure can be specially calculated as in Equation (4.9). For both equations, p is the precision and r is the recall of the classifier.

$$\mathrm{F}_\beta = \frac{(1+\beta^2)(pr)}{\beta^2 p + r} \tag{4.8}$$

$$\mathrm{F}_1 = \frac{2pr}{p + r} \tag{4.9}$$

While each of the precision, recall, and F-measures do better quantify the performance of a classifier on the minority class, they still do not address what happens when a trained model is the trivial majority classifier. In such a case, all of these measures would equal 0 because there would be no true positives as a result of the classifier predicting everything negative. These evaluation statistics are still parameterized based on the default threshold of 0 for the discriminant values. We must use different evaluation statistics that don't care about the threshold selected by the classifier to quantify its strength. There are two candidates: the ROC and the P-R curve.

Both the ROC and the P-R curve measure the overall goodness of a classifier across all possible discrimination thresholds between the two classes. Classifiers give not only a classification for a sample, but also a quantity representing how confident the algorithm is of these results. Using the confidence values for the samples, we can calculate statistics using the discrimination threshold between each pair of samples. The ROC calculates the true positive and false positive rates, and the P-R curve calculated the precision and recall. These pairs of rates can then be plotted to form the ROC and P-R curve respectively. Using the raw curves to choose the proper threshold for best classification is difficult, but we can use the curve to get one number that quantifies the strength of a classifier. When we integrate across the full range of the false positive rate or recall (from 0 to 1), the result is the area under the curve. The area under these curves are the two evaluation statistics we used to evaluate classifiers; we designate the area under the ROC as AUC, and the area under the P-R curve is PRC. ROC evaluation has been used by a number of researchers since the first Workshop on Imbalanced Data Sets at the AAAI conference in 2000 [41], but P-R curve evaluation is relatively new [21]. An example of equivalent ROC and P-R curves is shown in Figure 4.1.



(a) ROC

(b) PRC

Figure 4.1. Example of equivalent ROC and P-R Curves

Table 4.1. Data Sets

Name	Samples	Features	Samples by Class	Domain
CNS1	60	7129	21 39	BIO
CNS2	95	7129	28 67	BIO
LYMPH	77	7129	19 26 32	BIO
LEUK	72	7129	9 25 38	BIO
OVARY1	116	9200	16 100	BIO
OVARY2	116	9200	16 100	BIO
PROST1	216	9200	26 190	BIO
PROST2	233	9200	43 190	BIO
PROST3	253	9200	63 190	BIO
NIPS	896	7809	45 48 48 70 71 97 97 155 265	TEXT
SCR	540	3085	60 60 60 60 60 60 60 60 60	SCR

4.3 Data Sets

Imbalanced data sets show up in many different domains. Many papers have explicitly looked at imbalanced data in single domains, but we have not seen research that experiments on imbalanced data sets from vastly different domains in the same framework. Thus, we used data sets from multiple types of sources to show the effectiveness of feature selection metrics on all kinds of imbalanced data. These sources include microarray and mass spectrometry (CNS, LYMPH, LEUK, OVARY, PROST), text mining (NIPS), and spoken character recognition (SCR) data sets. The data sets we examined are summarized in Table 4.1. While there is only one data set for the text mining and character recognition domains, they are each divided into nine one-vs-rest problems for this study. Thus, there are 11 problems in the biological domain, 9 in the text domain, and 9 in the character recognition domain. In the NIPS data set, we discarded those features that were present in less than 5 samples; that left us with the 7809 features stated. No other data sets discarded features. All of the data sets are publicly available on the author's website.

4.4 Problem Frameworks

We aim to answer six different questions over the course of this paper:

1. *Which feature selection metrics perform best on average?* The performance of a feature selection metric depends on the nature of the problem. Some problems may have equal misclassification costs; some problems may want to maximize the ratio of true positives to those samples marked positive; others may simply look for the highest proportion of true positive samples. To that end, we compared each of the seven feature selection metrics using AUC and PRC. These feature selection metrics were tested by training the SVM using 10, 20, 50, 100, 200, 500, and 1000 features. Because many of the data sets are too small to be split into training and testing partitions, we evaluated each metric using 4-fold stratified cross-validation repeated with five different sets of folds for each data set. We also compared the feature selection metrics to the baseline performance of a linear SVM using all features to show the gains in performance from using feature selection.

2. *Which feature selection metrics are most likely to perform best for a single data set?* Most data mining researchers would not care if a learning method gave great results on average if it did not give good results on their problem of interest. Thus, we need a new framework to better answer this question. Forman [32] proposed an analysis technique to compare various methods on a large number of data sets. First, for each data set, we take the best score achieved by all of the methods for an evaluation statistic. Then, for each method, we find the ratio of data sets for which this method gave scores within a certain tolerance of the best score for each data set. For example,

if we allow a tolerance of 1% for a best score of 0.90, 0.895 would be a success and 0.885 would be a failure. Those feature selection metrics with the highest ratios are considered the closest to optimal. We performed this test using the results from the above framework with 10 features selected and with the optimal number of features selected.

3. *Which feature selection metrics perform best for different domains?* Imbalanced data sets come from vastly different applications, and the goals guiding a machine learning approach to these tasks often differ based on the inherent characteristics of their data sets. We divided these problems into the different areas (biological data analysis, text mining, and character recognition) and analyzed each feature selection metric on these subsets of problems like in the first problem framework.

4. *How does feature selection compare with sampling techniques and algorithms on the class imbalance problem?* As we stated in the introduction, no research has compared the feature selection, resampling, and algorithms approaches to the class imbalance problem. We would like to see whether one of these approaches is the best or if they all perform equally well. We compared the performance of the best feature selection metric from the first framework with the SMOTE/random under-sampling hybrid and AUC maximizing SVM approaches using each of the evaluation statistics.

5. *Can the different approaches be used together to get even more improved results?* Numerous studies have shown that the sampling, algorithms, and feature selection methods can improve the performance of a classifier on an imbalanced data set. Very little research has been conducted on whether

these methods can be combined. The only study we are aware of is by Al-Shahib, Breitling, and Gilbert [2], where they combined wrapper-based feature selection and full random under-sampling. They showed that feature selection and under-sampling were both important, but the most important factor was the under-sampling. Because each of these approaches to the class imbalance problem manages the challenges in a different way, they could result in a greater performance than any individual part could on its own. We used the best feature selection metric from the first framework, the SMOTE/random under-sampling hybrid, and the AUC maximizing SVM together and compared it with the results of each component individually.

6. *Which feature selection metrics perform best regardless of the classifier used?* Some feature selection metrics work very well with specific learning methods. OR helps the naïve Bayes classifier achieve the best result possible [62]. RELIEF was designed based on a nearest neighbor philosophy [47, 50] and gives the nearest neighbor more improvement than simple correlation coefficients [18]. C4.5 and other decision tree algorithms intrinsically use IG as their node-splitting statistic. Finding a feature selection metric that performs well across different induction philosophies would make a data mining researcher's work much easier. Thus, using the framework from the first problem, we evaluated the feature selection metrics on the nearest neighbor and naïve Bayes classifiers as well. We then took the mean of the performance of each classifier on each evaluation statistic to compare between feature selection metrics. Those metrics with the highest performance across different classifiers are judged to select the best features.

Chapter 5

Results

The experimental results are provided in this section. All experiments were conducted using the MATLAB SPIDER package and CLOP extensions with the following exceptions. Each of the feature selection metrics except for PCC and S2N were custom implementations by the authors. The nearest neighbor algorithm was modified to allow for AUC and PRC analysis based on the difference between the nearest neighbor and the nearest neighbor from a different class than the closest point. The SVM^{perf} algorithm was used instead of the regular SVM using the MEX interface and SPIDER objects developed by Oscar Luaces [54, 55]. All statistical tests used in comparing different methods were the paired T-test.

5.1 Best Average Performance

Figures 5.1 and 5.2 show the average AUC and PRC for the SVM across the different feature selection metrics as the number of features varies. Across all possible feature counts, the top performing metrics on average are IG, S2N, and FAST. These metrics are consistently close to the best average performance across each of the evaluation statistics with FAST being the best with less features selected and S2N or IG being the best with more features selected. Any of these three metrics would be a good choice for use on an arbitrary imbalanced data set.

There are three main trends we see in these graphs. First, regardless of the evaluation statistic used, FAST performs best by a decent margin for only 10 features selected. This margin is statistically significant at the $\alpha = 0.05$ significance level. This is important because while we would like to see the best predictive power using feature selection, there are many domains where selecting a large number of features would make later analysis of the selected features impossible. For example, in biological data analysis, we would like to have a list of genes that influence whether a person has cancer or not. A biologist would likely prefer to see this list be as small as possible while still getting good results. This goal allows us to attain data savings as each of our tested data sets will be shrunk to a fraction of their original size.

We also see a trend in these data sets that 50 features appears to be the point where the best performing feature selection metrics peak across each evaluation statistic. With more than 50 features selected, these metrics see a significant decline in performance. The goal of data mining is to make the best predictions possible; on high dimensional imbalanced data sets, it appears that we only need to select between 50 and 100 features to attain this peak performance. At the same time, we still see data savings with feature selection where our tested data sets would be shrunk to hundredths of their previous size.

Finally, except for OR, every single feature selection metric beat the baseline performance for 50 and more features selected; when we discount FAIR as well, the remaining feature selection metrics equalled or bettered the baseline performance with just 10 features selected. When using the linear SVM, not only will we see data savings from extensive feature seelction, but we can see significant improvements in performance as well. Feature selection is a good approach to manage the class imbalance problem on high dimensional data sets when using the SVM.

5.2 Probability of Best Performance

Figures 5.3 and 5.4 show the ratio of data sets for which each metric is within a small percentage of the best score achieved by any feature selection metric for AUC and PRC with 10 features selected. Figures 5.5 and 5.6 are similarly shown for 50 features selected. We chose 10 features to show the effectiveness of the initial features selected by each metric, and we chose 50 features as this is the level of best performance from the previous framework.

With only 10 features selected by each metric, FAST is clearly the best metric. FAST outperformed every other metric at the 1% tolerance level by nearly 10% for both AUC and PRC. Other than PCC at the $3-5\%$ tolerance levels for PRC, FAST had the highest proportion of successes by a significant margin. Researchers working in domains that desire lower feature counts for various reasons like interpretability of the features selected should use FAST to have the best chances of getting strong performance on their data.

When looking at the first 50 features selected, we have some trouble deciding if there is a best feature selection choice. At the highest amount of tolerance allowed, there is very little discernibility between CHI, PCC, S2N, IG, and FAST for each evaluation statistic. The marginally best odds most frequently come from the IG and S2N feature selection metrics for tolerance levels at 2% and above.

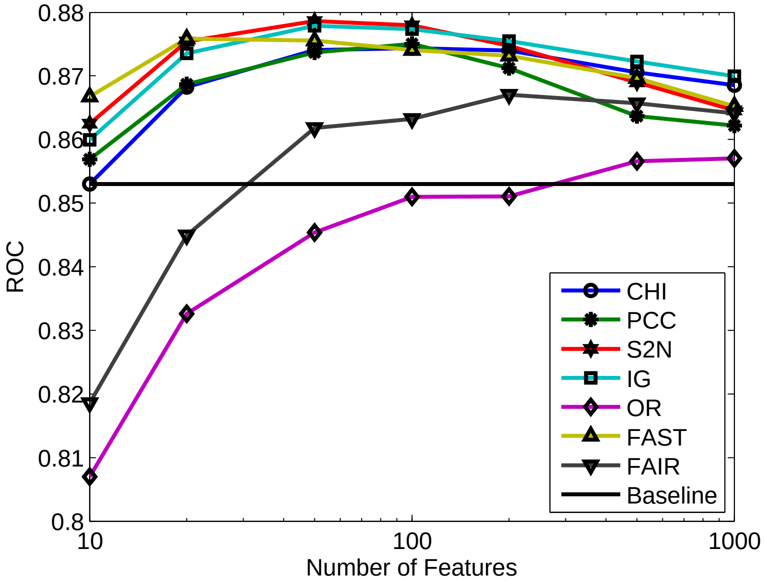


Figure 5.1. Average AUC Performance

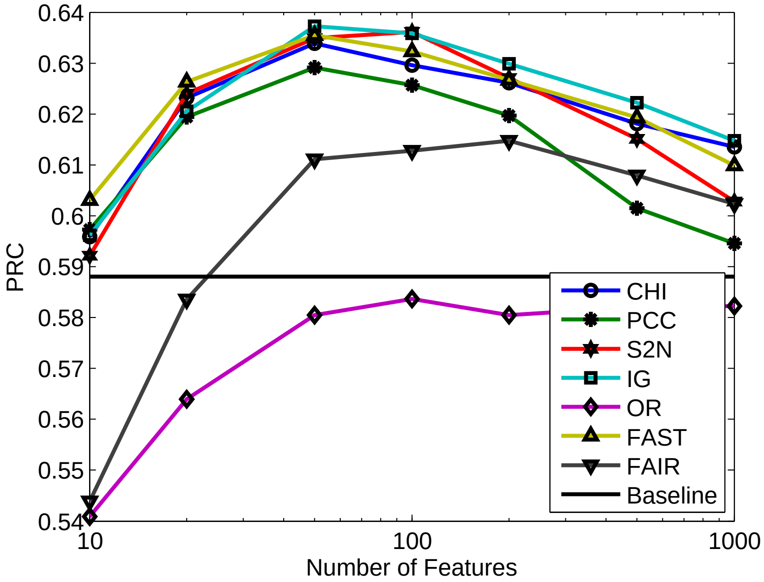


Figure 5.2. Average PRC Performance

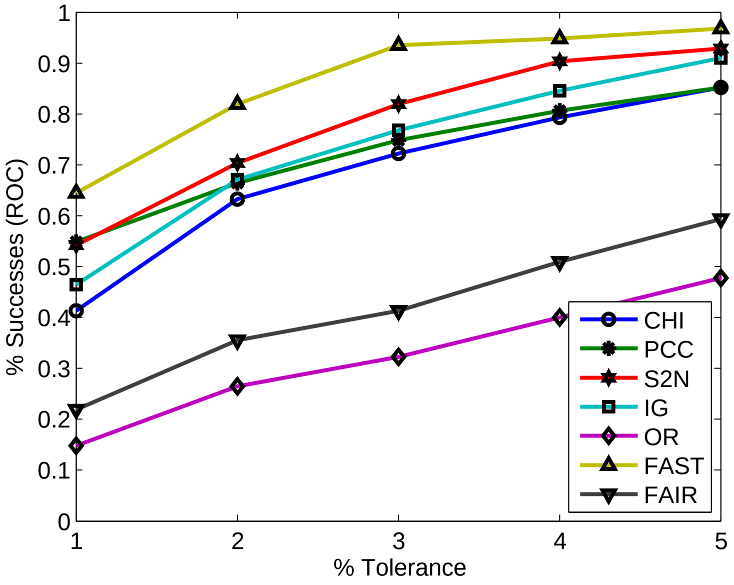


Figure 5.3. Percent of Problems Metrics Performed Within Tolerance of Best Metric, AUC, 10 Features

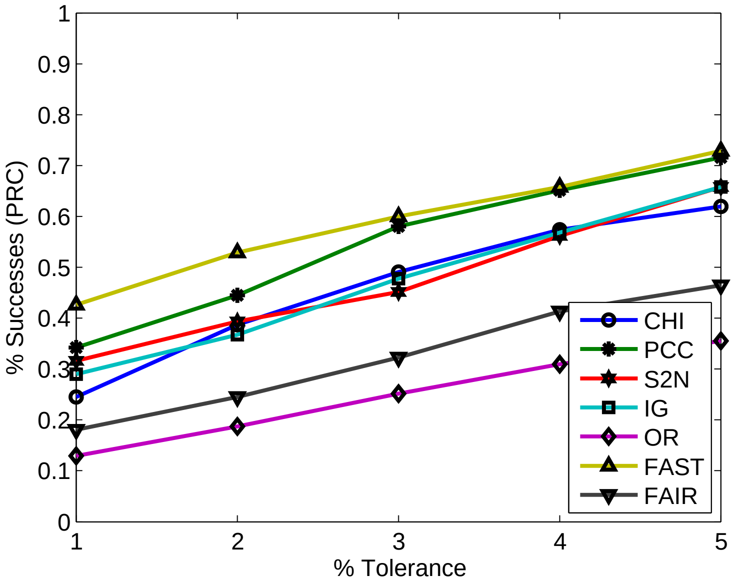


Figure 5.4. As Figure 5.3, for PRC

When we allow only 1% tolerance, FAST is virtually indistinguishable in odds compared to S2N and IG. Most data mining researchers would prefer the smallest tolerance possible, so the 1% level is the most important. Thus, we believe that for the odds of the best performance possible, any of the IG, S2N, and FAST metrics would be strong choices.

5.3 Domain Analysis

Figures 5.7 and 5.8 show the average AUC and PRC graphs for the biological data sets. Similarly, Figures 5.9 and 5.10 show the results from the text mining problems and Figures 5.11 and 5.12 show the results from the character recognition problems.

For the biological data sets, there are two conflicting goals: best performance possible, and performance with fewest features for biological analysis. At 1000 features, IG and CHI have the best performance (though not statistically the best), and OR and FAST tie for the best with AUC. With only 10 features, FAST is the clear winner as it performs statistically better than every other feature selection metric for both AUC and PRC scores at a significance level of $\alpha = 0.02$ except for the IG-AUC combination. This jump in performance can be attributed to FAST selecting features that better spread out the means of the two classes compared to the first features selected by other metrics.

For the text mining data sets, the big result is that the feature selection metrics did not improve performance over the baseline. This appears to be counter to prior research [32, 62, 78], but those results only found that a few metrics improved performance. The primary difference between the NIPS data set we used and the many data sets experimented on in the previous work on text data is the

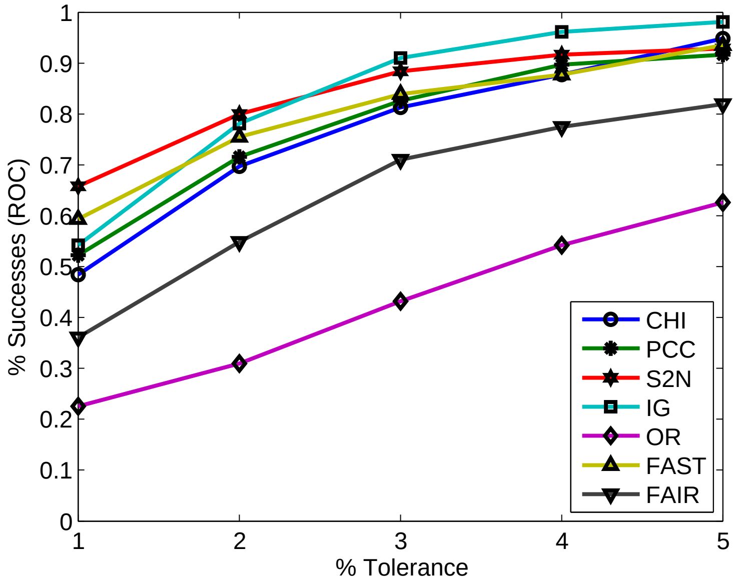


Figure 5.5. Percent of Problems Metrics Performed Within Tolerance of Best Metric, AUC, 50 Features

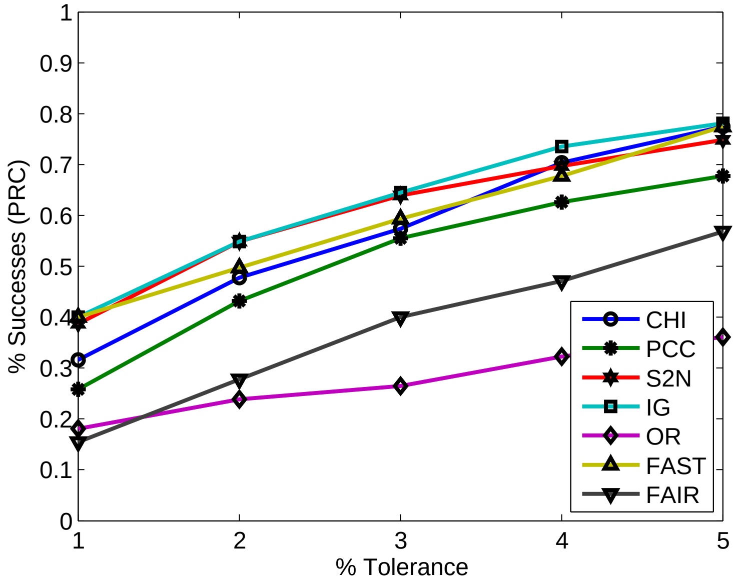


Figure 5.6. As Figure 5.5, for PRC

feature representation. The data sets tested by the previous authors had Boolean features; in contrast, the NIPS data set contains the raw word counts. The features in a Boolean bag-of-words data set are typically only weakly indicative of the class [24]; we believe that using the raw word count makes these features even more weakly indicative of the class. The feature representation in sparse data sets is extremely important. The discrete nature of the features explains why IG and CHI perform the best on both AUC and PRC: the difference between a word appearing infrequently versus a word showing up frequently is much greater than a word showing up exactly four or exactly five times.

With the spoken character recognition problems, we prefer a small number of features, but for a different reason than the biological analysis sets: the vast majority of features available contribute nothing but noise to the learning algorithm. Thus, we are not constrained to selecting just 10 features. With that in mind, the best feature selection metrics are S2N, FAST, and PCC. These metrics do not discretize the features; converting noisy continuous features into binary features appears to amplify the noise and makes IG and CHI perform worse with a small number of features. In contrast, the continuous metrics were not affected by the noise as much and separated the classes slightly better.

5.4 Analysis of Different Approaches

Figure 5.13 compares the S2N algorithm results using 50 features with those achieved using the SMOTE/random under-sampling hybrid (SMUS) and the AUC maximizing SVM (AUCSVM) on AUC and PRC. This figure also includes information about three combination approaches: using S2N with 50 features on the AUCSVM (F/A), using the sampling technique and AUCSVM (S/A), and using

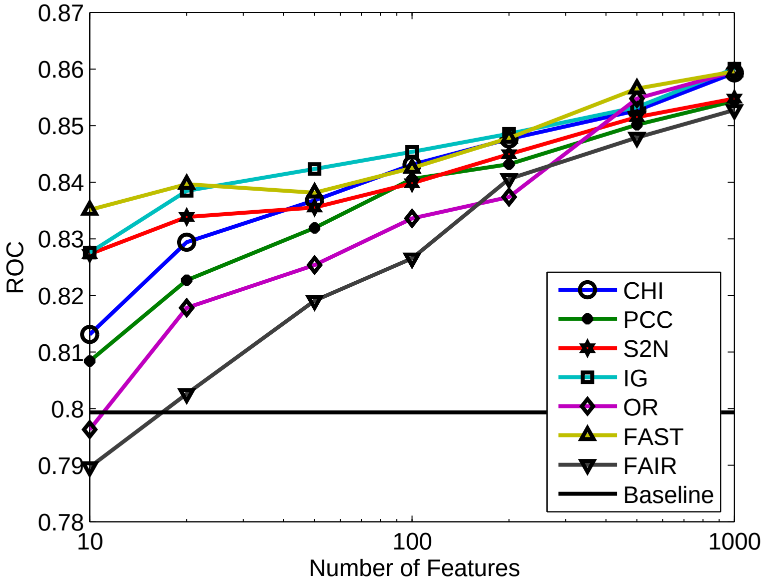


Figure 5.7. Average AUC Performance on Biological Analysis Data Sets

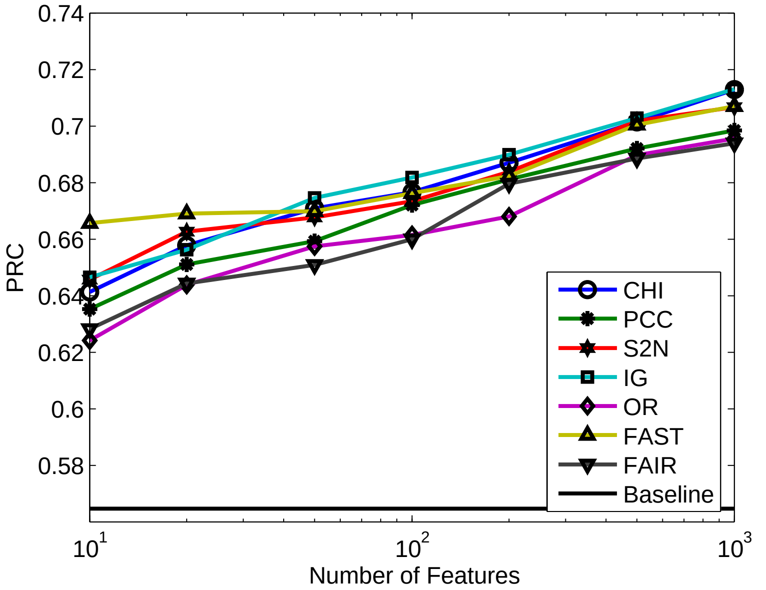


Figure 5.8. Average PRC Performance on Biological Analysis Data Sets

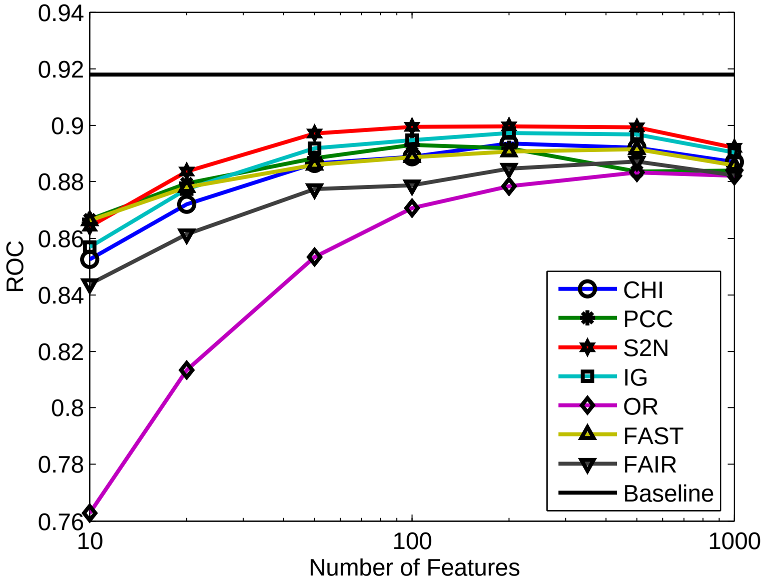


Figure 5.9. Average AUC Performance on Text Mining Data Sets

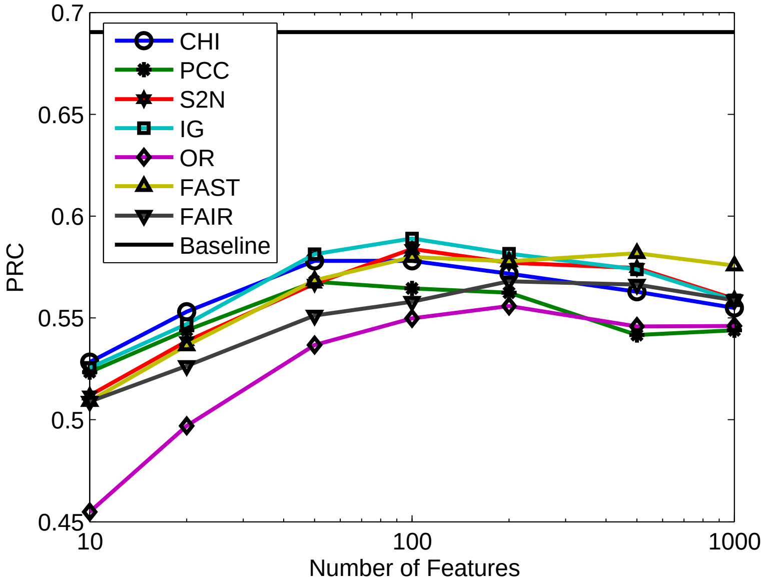


Figure 5.10. Average PRC Performance on Text Mining Data Sets

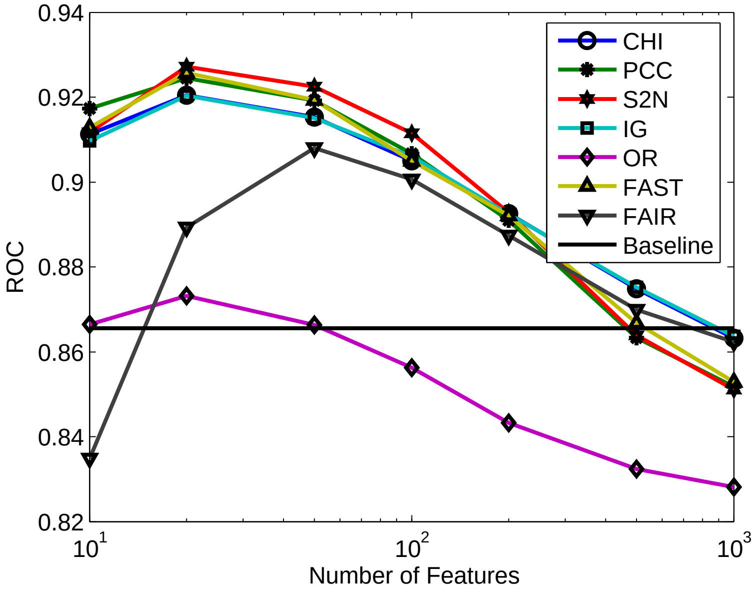


Figure 5.11. Average AUC Performance on Character Recognition Data Sets

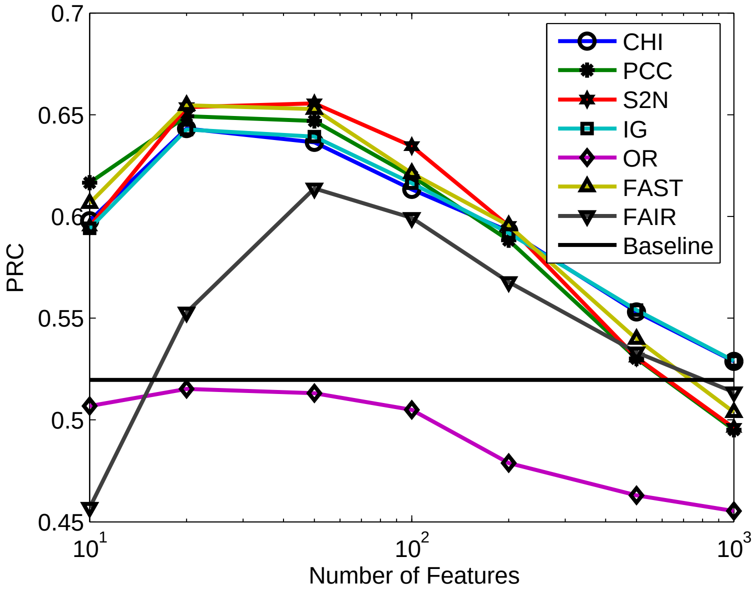


Figure 5.12. Average PRC Performance on Character Recognition Data Sets

all three approaches at once (ALL). The best results were seen using S2N alone. The next best performers are the AUCSVM and the F/A hybrid technique. All three methods using the sampling technique lagged far behind with ALL being the worst. In fact, the performance of the resampling methods is below the baseline with all features.

The good performance of the AUCSVM and the F/A hybrid is not surprising given the results from many researchers [7, 46] that the AUCSVM performs better than the standard SVM. However, it is slightly surprising that using feature selection with the AUCSVM did not improve on this performance at all. Based on the results from Forman [33], the AUCSVM is a very strong algorithm that may see poor performance with feature selection compared to weaker algorithms like the naïve Bayes classifier.

The most interesting results from Figure 5.13 are that the methods using resampling performed poorly, with the ALL hybrid performing about 5% below the baseline. Why did we see these results? Manevitz and Yousef [58] believe that a researcher should not expect as good of results from only positive data when negative data is available, and Elkan [29] says that even converting negative samples to unlabeled samples will hinder the performance of a classifier. Thus, the random under-sampling approach, which discards information about the majority class, is eliminating valuable information from the data set. It appears that using synthetically created minority samples does not avoid this issue. The resampling hybrid methods suffer from the additional problem of matching the training distribution too closely like the F/A hybrid, and combining all three makes for a real strong algorithm that overfits the training data.

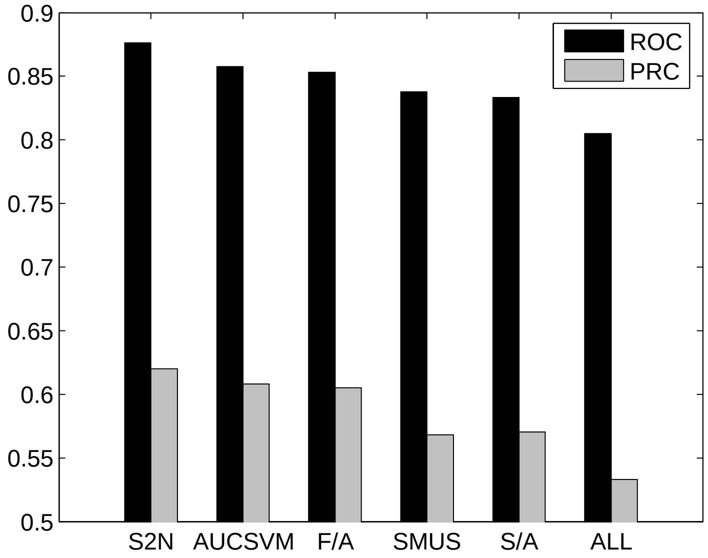


Figure 5.13. Average Performance of Different Canonical Approaches

5.5 Feature Selection Metric Effectiveness

Figures 5.14 and 5.15 show the mean of the average performance for the SVM, nearest neighbor, and naïve Bayes classifiers over each evaluation statistic as we vary the number of features. As well, Figures 5.16 and 5.17 show the average performance for the nearest neighbor algorithm using AUC and PRC respectively, and Figures 5.18 and 5.19 show the average performance for the naïve Bayes algorithm. Recall that we are interested in showing which feature selection metrics perform best regardless of the classifier a researcher may use.

The first clear trend is that the average performance of each classifier with feature selection is better than the average baseline performance of each classifier. This is not a surprising result. Feature selection helped the SVM improve, and most researchers believe that the SVM is relatively insensitive to feature selection. In contrast, the nearest neighbor and naïve Bayes classifiers are very sensitive to feature selection, and this is why the best gains over the baseline are larger than those seen in Figures 5.1 and 5.2 with just the SVM classifier compared to Figures 5.16–5.19.

The second trend is that there are two feature selection metrics that are most effective, but in different circumstances. When selecting between 10 and 50 features, FAST is the best performer, and S2N is the second best. When selecting 100 or more features, S2N is the most effective; FAST is the second most effective until we select 1000 features. FAST being the most effective with a small number of features selected is not very surprising given that it resulted in the biggest boosts at 10 features of any of the tested metrics. Likewise, it's not surprising that S2N is effective at higher feature counts because its performance on the SVM was near the best. Thus, depending on the number of features desired, FAST or S2N would be a good feature selection metric to use regardless of the classifier choice.

Third, there is a significant drop in performance occuring for each feature selection metric across all evaluation statistics. This drop is large enough that PCC and OR are actually regressing beyond the baseline performance. Additionally, every single feature selection metric except for FAIR performed worse when selecting 1000 features than when selecting just 10 features; some of these drops in performance occur as early as 100 features selected. It is vital to choose the number of selected features carefully. With too few features, an algorithm's performance may not be as good as possible. With too many features, you are more likely to see overfitting in your classifier. We recommend starting with 50 features as a baseline and empirically comparing test results from different feature counts to find the best feature count.

Finally, compare Figure 5.1 with Figure 5.18. When using feature selection, the performance of the linear SVM is actually substandard to the performance of the naïve Bayes algorithm. The best results at 10 and 20 features selected are found using the naïve Bayes classifier with FAST features. This result follows the findings of Forman [33] on how different classifiers respond to different distributions of data. He found that the SVM gave the best results when the classification task was mostly balanced, but when the data set had a very small number of samples in the minority class, like the data sets we studied here, or a rather extreme skew between the two classes, the naïve Bayes and multinomial naïve Bayes algorithms performed the best. If the goal is to make the best generalizations possible, the simplest algorithms are often the best choice.

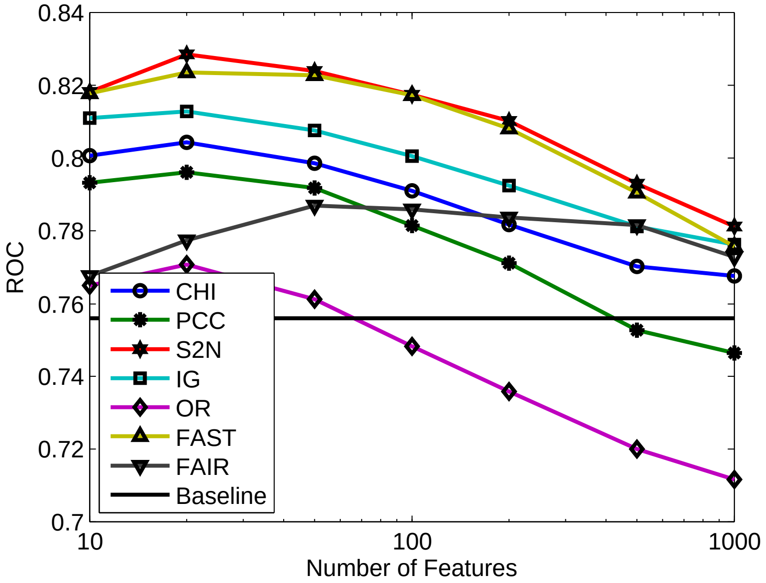


Figure 5.14. Average AUC Performance, Mean of Multiple Classifiers

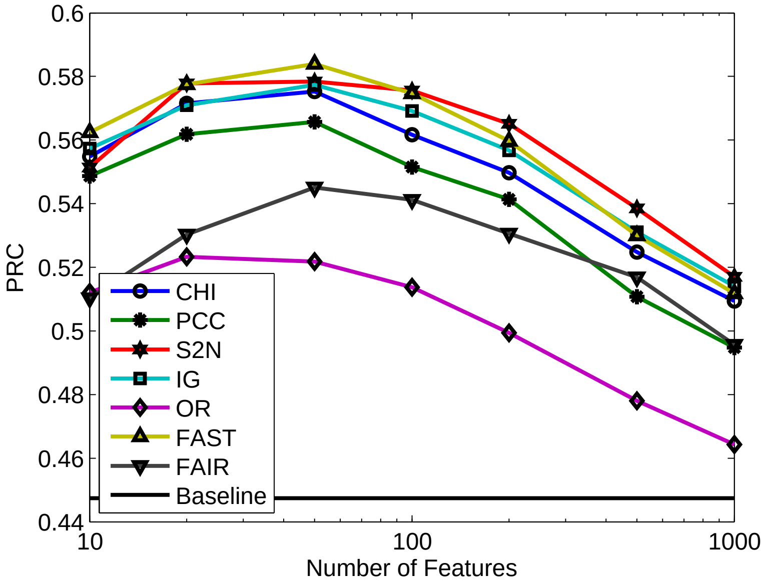


Figure 5.15. Average PRC Performance, Mean of Multiple Classifiers

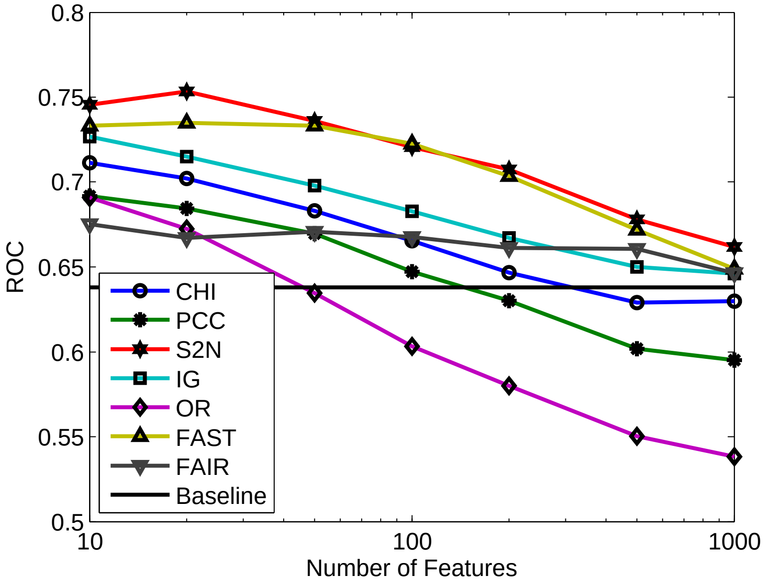


Figure 5.16. Average AUC Performance, Nearest Neighbor

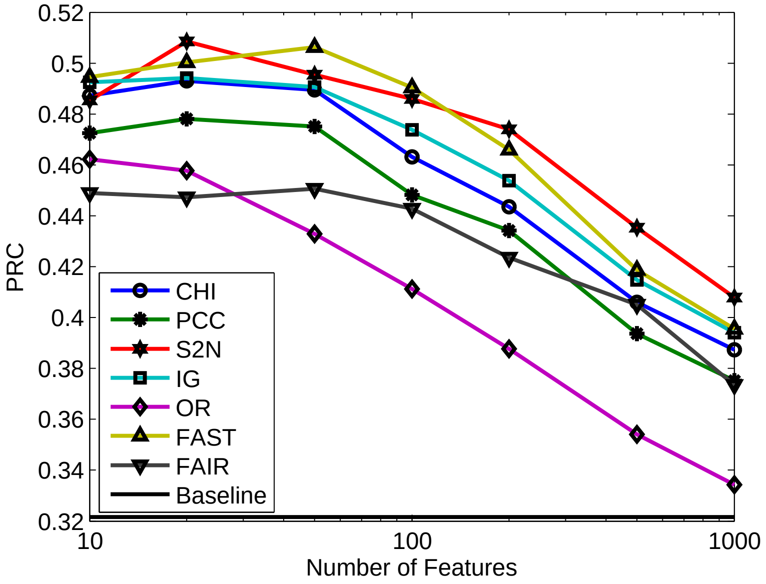


Figure 5.17. Average PRC Performance, Nearest Neighbor

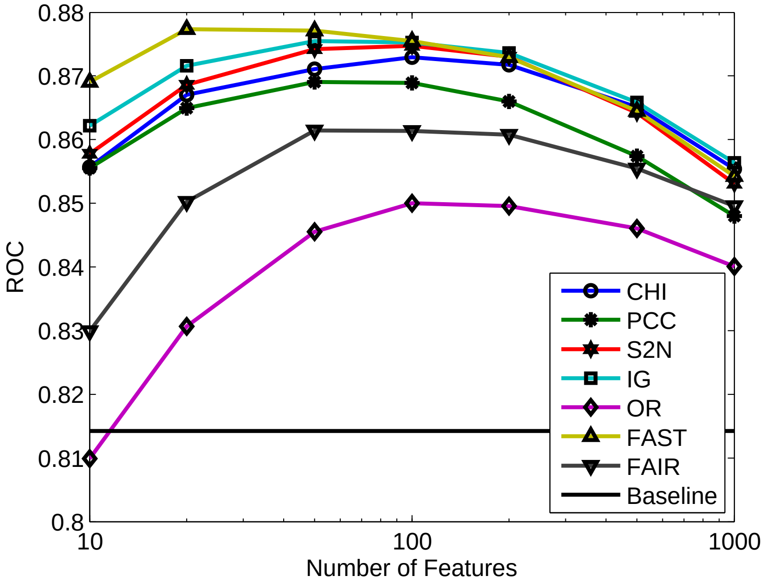


Figure 5.18. Average AUC Performance, Naïve Bayes

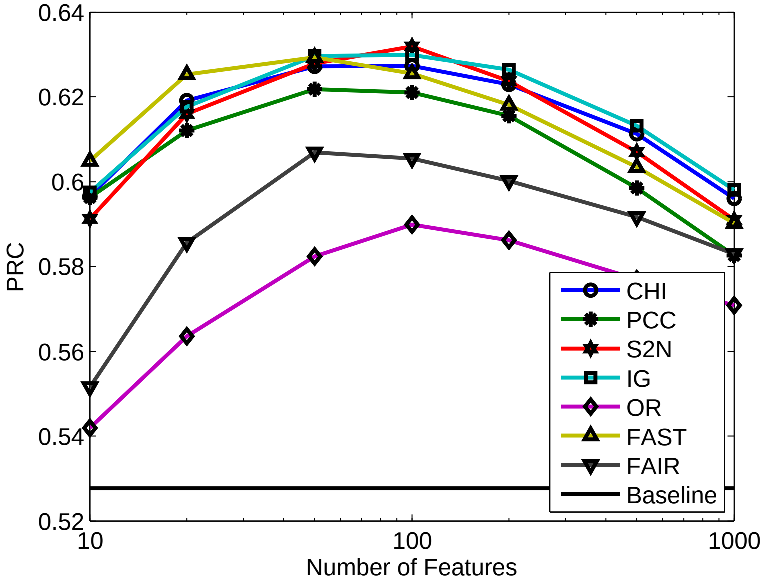


Figure 5.19. Average PRC Performance, Naïve Bayes

Chapter 6

Concluding Remarks

Machine learning and data mining techniques have problems handling imbalanced data sets that frequently arise from real-world applications. Researchers have developed many new methods for combating the class imbalance problem. Feature selection has been used frequently to reduce the dimensionality of a data set, but its merit in imbalanced problems has only been recently studied.

6.1 Contributions

We developed a new feature selection metric, FAST, that evaluates features using the area under an approximate ROC to get a non-parametric view of the predictive power of a feature to the classification task. To speed up the calculation of the AUC, we used a modified histogram, called an even-bin distribution, that fixes the number of points in each bin and varies the width of the bins appropriately. We select one threshold for each bin; this threshold is set to the mean feature value of the points in each bin. We also developed a comparable algorithm, FAIR, that uses the approximate P-R curve as its evaluation statistic.

We also conducted the first systematic study of methods from each of the three types of approaches to handling imbalanced data sets. Previous comprehensive studies solely examined resampling methods [40], feature selection methods [32], and new algorithms [46,58,59] without considering whether other types of methods might perform better. In contrast, this study compared a number of feature selection methods, the SMOTE and random under-sampling hybrd, and the AUC maximizing SVM on many small sample imbalanced data sets from a variety of applications.

6.2 Conclusions

The biggest finding from this paper is that feature selection is beneficial to handling most high dimensional imbalanced data sets. This holds across the biological and character recognition domains, and it also holds for each of the seven feature selection metrics we tested. For the text mining domain, other research [32, 62, 78] have found improved results on other data sets with feature selection. We believe that this happens because the original data sets provide samples that are too disparate to make generalizations about groups of the samples. When we reduce the number of features in the data set to approximately the same order of magnitude as the samples, we see a significant jump in performance regardless of the classifier used. At this point, each of the induction algorithms tested could make generalizations about different clusters of points. Thus, feature selection is paramount to achieving the best possible results in learning from high dimensional imbalanced data.

Between the seven different feature selection metrics investigated, two metrics were generally the best performing across each of the tests: FAST and S2N. It is not surprising that these two metrics would perform nearly equally well as they both try to maximize the spread between the classes through different calculations. These differences make the metrics perform better in different situations. FAST is the best metric when selecting very small numbers of features, especially just 10 features. S2N is the best metric when selecting more than 50 features. These two metrics are the best on every classifier tested.

6.3 Future Work

There is still a lot of room for improvement. The best average PRC score achieved in this experiment was under 0.6. For a problem like determining whether somebody has cancer based on their gene profile, this level of precision is extremely small and useless as a test. If we aim to learn from small samples, we need to find ways to mitigate this problem even further. One potential path of future research would be to use information from related problems to guide the construction of a kernel for an SVM after feature selection has been used. The KernelBoost algorithm is an implementation of this idea that can work on semi-supervised and unsupervised data sets; thus, we would allow the use of unlabeled data in building a better classifier [38].

Improvement may also be found by using more optimal feature selection methods. Koller and Sahami [49] showed that the optimal feature ranking can be calculated using Markov blankets in a backward substitution algorithm; Fleuret [31] and Peng, Long and Ding [64] derived two approximations to this optimal ranking. However, they are forward-selecting methods that use the previously selected features to guide their search for the next feature to add to the feature set. Additionally, these methods were not designed to handle imbalanced data. If we try to find the Markov blanket ranking of features for an imbalanced data set with high dimensionality, there is no guarantee that any method will find this best feature ranking without performing a brute force search or using a large number of samples. The brute force search makes these methods intractable, and the sample size requirement makes the use of these methods infeasible for the data sets we examined. All of the metrics investigated in this study have been used to varying degrees of success on imbalanced data. Future research should compare the performance of more optimal feature ranking methods with standard feature selection metrics as there may still be some benefit to using forward-selecting methods.

One interesting result from this study is that resampling did not improve performance at all. Clearly, this should not be taken as evidence that the resampling approaches do not work. The survey of resampling techniques by Van Hulse [40] showed that almost every resampling technique they investigated resulted in some improvement over no resampling. However, the data sets he examined varied in size from just over 200 samples to the tens of thousands. It appears that resampling approaches work better on larger data sets; future research should examine this issue specifically.

Additionally, we found that the AUC maximizing SVM did not improve performance at all. According to Forman's analysis of learning from small samples [33], this may be an issue with the use of stratified cross-validation rather than any condemnation of the resampling and algorithm approaches. Most of the learning algorithms researchers use today were originally designed and tested on large, fully balanced data sets. Algorithms trained with real-world data sets must be able to handle potentially large mismatches between the training and test distributions; the development of algorithms that are less distribution-specific is an open research problem.

One issue regarding the generalizability of these results to imbalanced data sets in general is that many are not small samples, many others are not high dimensional, and still others have very large imbalances between the classes. In these cases, the results found in this experiment do not hold. The primary strength of feature selection metrics, that they are linear with respect to the feature dimensionality, is negated when dealing with small feature sets. Different data sets that are larger, have fewer features, or have higher skew, likely play more to the strengths of algorithms and resampling methods. Varying levels of noise may also affect the performance of each class of approaches. Future research should explicitly compare and contrast the size, skew, noise, and dimensionality of imbalanced data sets to determine the influence each of these characteristics has on the performance of resampling methods, algorithms, and feature selection methods.

A related problem to high dimensional imbalanced data that has not been extensively studied is learning from heterogeneous data. Heterogeneous data sets have features drawn from different sources. For example, image classification tasks can use cameras that detect different wavelengths of light, and bioinformatics tasks can exploit microarray RNA, messenger RNA, and gene ontology information. The use of heterogeneous data allows a researcher to leverage different characterisitics of a sample that one homogeneous data set cannot represent. As more heterogeneous data sets are used in a classification task, the number of features will dramatically increase. This jump in feature set size means that feature selection is even more necessary with heterogeneous data sets. The problem is naïvely combining all of the features together will not work as this may cause an algorithm to overfit the data. Feature selection metrics must integrate the information in each data set to select the best feature set [77]. Researchers should examine feature selection methods on imbalanced, heterogeneous data sets to seek the best classification possible.

Terms

Accuracy Ratio of the sum of true positives and true negatives to the number of predictions made. 1–4, 14, 17, 20–23, 30, 52

AUC Area under the ROC. ii, 3, 6, 22, 23, 32, 33, 40–42, 44, 54, 57–60, 62, 65, 67, 72, 78, 79, 81

Bagging Ensemble that trains base classifiers using random samples, with replacement, from the training data set. 4, 17, 19, 21, 22, 44

Bayes optimal classifier Ideal ensemble containing all possible hypotheses. 24

Bi-normal separation A feature selection metric based on the difference between the standard normal CDFs at the true positive rate and false positive rate, abbreviated as BNS. 30, 31

Boosting Ensemble that trains base classifiers in a series using weighted samples, with replacement, from the training data set. 4, 17–20, 22, 44

Chi-square statistic A feature selection metric that measures the independence of a feature from the class labels, abbreviated as CHI. 31, 36, 37, 62, 65, 67

Class imbalance problem An issue with machine learning where an algorithm performs well on majority samples but performs poorly on minority samples. ii, 1, 2, 4–6, 8, 10, 14, 17, 19, 23, 26, 30, 34, 44, 45, 62, 78

Cost-sensitive learning Learning when different misclassification errors have different costs. 2, 4, 11, 18, 20–22, 24, 44

F-measure Weighted harmonic mean of precision and recall. 3, 30, 52, 53

F1-measure Specific instance of the F-measure where precision and recall are evenly weighted. 20, 23, 30, 34, 53

Feature Assessment by Information Retrieval A modification of the FAST metric that evaluates features using area under the P-R curve, abbreviated as FAIR. 42, 61, 73, 78

Feature Assessment by Sliding Thresholds A feature selection metric that evaluates features using area under the ROC, abbreviated as FAST. ii, 6, 32, 33, 42, 60–62, 65, 67, 73, 74, 78–80

Information gain A feature selection metric that measures the difference between the entropy of the class labels and the conditional entropy of the class labels given the features values, abbreviated as IG. 31, 36, 37, 59–62, 65, 67

Naïve Bayes Learning method that assumes full conditional independence of its features and maximizes the posterior probability of its predictions. 2, 5, 30, 46, 49–51, 59, 71, 72, 74

Nearest neighbor Learning method that classifies samples into the class of the nearest training sample. 3, 11, 12, 24, 28, 46, 50, 51, 59, 72

Odds ratio A feature selection metric that measures the odds of a feature occuring in one class normalized by the odds of the feature occuring in the other class, abbreviated as OR.. 30, 31, 37, 38, 46, 59, 61, 65, 73

One-class learning Learning using samples drawn solely from one class. 4, 14–16, 44

Pearson correlation coefficient A feature selection metric that measures the strength and degree of the linear relationship between a feature and the class labels, abbreviated as PCC.. 32, 38, 39, 60, 62, 67, 73

PRC Area under the P-R curve. ii, 33, 42, 54, 57, 60, 62, 65, 67, 72, 80

Precision Ratio of the true positive count to the number of positive predictions made. 20, 23, 30, 42, 52–54, 80

Precision-recall curve A comparison of the precision and recall of a classifier across all possible decision boundaries, abbreviated as P-R curve. 32, 33, 42, 54, 78

Recall Ratio of the true positive count to the number of positive samples. 20, 23, 30, 35, 42, 52–54

Receiver operating characteristic A comparison of the true positive rate and false positive rate of a classifier across all possible decision boundaries, abbreviated as ROC. 32, 33, 40, 42, 54, 78

RELIEF A feature selection metric that scores based on the distance of samples to the nearest sample in the same class and the nearest sample in the other class. 28, 29, 32, 46, 59

Signal-to-noise correlation coefficient A feature selection metric that evaluates features by the ratio of the distance between the class means to the sum of the standard deviations of the classes, abbreviated as S2N. ii, 39, 60–62, 65, 67, 71, 73, 79, 80

Small sample problem An issue with machine learning where an algorithm does not have enough training data to generalize patterns from the data. ii, 2, 5, 6, 10, 26, 79–82

Support vector machine Learning method that finds the maximum-margin hyperplane, abbreviated as SVM. 2, 5, 15, 16, 22, 23, 26, 29, 30, 32, 44, 46, 48, 51, 57–60, 62, 67, 71–74, 79–81

Trivial majority classifier Classifier that naïvely assigns every sample to the majority class without examining the feature values. 3, 9, 20, 22–24, 44, 52, 54

References

[1] N. Abe, B. Zadrozny, and J. Langford, “An iterative method for multi-class cost-sensitive learning,” in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 204, pp. 3–11.

[2] A. Al-Shahib, R. Breitling, and D. Gilbert, “Feature selection and the class imbalance problem in predicting protein function from sequence,” *Applied Bioinformatics*, vol. 4, pp. 195–203, 2005.

[3] E. Alpaydin, *Introduction to Machine Learning.* The MIT Press, 2004, pp. 43–45, 360–363.

[4] A. Arauzo-Azofra, J. M. Benítez, and J. L. Castro, “A feature set measure based on RELIEF,” in *Proceedings of the 5th International Conference on Recent Advances in Soft Computing RASC2004*, 2004, pp. 104–109.

[5] R. Barandela, R. M. Valdovinos, J. S. Sánchez, and F. J. Ferri, “The imbalanced training sample problem: Under or over sampling?” in *In Joint IAPR International Workshops on Structural, Syntactic, and Statistical Pattern Recognition (SSPR/SPR04)*, 2004, pp. 806–814.

[6] J. Bins, B. A. Draper, and F. D. Informtica, “Feature selection from huge feature sets,” 2001.

[7] U. Brefeld and T. Scheffer, “AUC maximizing support vector learning,” in *Proceedings of the ICML Workshop on ROC Analysis in Machine Learning*, 2005.

[8] L. Breiman, “Stacked regressions,” *Machine Learning*, vol. 24, pp. 49–64, 1996.

[9] ——, “Random forest,” *Machine Learning*, vol. 45, pp. 5–32, 2001.

[10] N. Chawla, K. Bowyer, L. Hall, and P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[11] N. Chawla, N. Japkowicz, and A. Kolcz, Eds., *Proceedings of the ICML’2003 Workshop on Learning from Imbalanced Data Sets.*, 2003.

[12] N. Chawla, A. Lazarevic, L. Hall, and K. Bowyer, “SMOTEBoost: Improving prediction of the minority class in boosting,” *Principles of Knowledge Discovery in Databases*, vol. LNAI 2838, pp. 107–119, 2003.

[13] N. Chawla, N. Japkowicz, and A. Kotcz, “Editorial: Special issue on learning from imbalanced data sets,” *SIGKDD Explorations*, vol. 6, no. 1, pp. 1–6, 2004.

[14] C. Chen, A. Liaw, and L. Breiman, “Using random forest to learn imbalanced data,” University of California, Berkeley, Tech. Rep., 2004.

[15] X. Chen, “An improved branch and bound algorithm for feature selection,” *Pattern Recognition Letters*, vol. 24, no. 12, pp. 1925–1933, 2003.

[16] X. Chen, B. Gerlach, and D. Casasent, "Pruning support vectors for imbalanced data classification," in *Proceedings of International Joint Conference on Neural Netowkrs*, 2005, pp. 1883–1888.

[17] X. Chen and J. C. Jeong, "Minimum reference set based feature selection for small sample classifications," in *Proceedings of the 24th International Conference on Machine Learning*, 2006, pp. 153–160.

[18] X. Chen and M. Wasikowski, "FAST: A ROC-based feature selection metric for small samples and imbalanced data classfication problems," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 124–133.

[19] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[20] Y. L. Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, 1990, pp. 598–605.

[21] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 30–38.

[22] P. V. der Putten and M. van Someren, "A bias-variance analysis of a real world learning problem: the CoIL challenge 2000," *Machine Learning*, vol. 57, no. 1-2, pp. 177–195, 2004.

[23] P. Domingos, "MetaCost: a general method for making classifiers cost-sensitive," in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999, pp. 155–164.

[24] M. Dredze, K. Crammer, and F. Pereira, “Confidence-weighted linear classification,” in *Proceedings of the 25th International Conference on Machine Learning*, 2008, pp. 264–271.

[25] C. Drummond and R. C. Holte, “Exploiting the cost (in)sensitivity of decision tree splitting criteria,” in *Proceedings of the 17th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 239–246.

[26] ——, “Severe class imbalance: Why better algorithms aren’t the answer,” in *Proceedings of the 18th European Conference on Machine Learning*, 2005, pp. 539–546.

[27] C. Elkan, “The foundations of cost-sensitive learning,” in *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001, pp. 973–978.

[28] ——, “Magical thinking in data mining: Lessons from CoIL challenge 2000,” in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 426–431.

[29] C. Elkan and K. Noto, “Learning classifiers from only positive and unlabeled data,” in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 213–220.

[30] A. Estabrooks, T. Jo, and N. Japkowicz, “A multiple resampling method for learning from imbalanced data sets,” *Computational Intelligence*, vol. 20, no. 1, pp. 18–36, 2004.

[31] F. Fleuret, "Fast binary feature selection with conditional mutual information," *Journal of Machine Learning Research*, vol. 5, pp. 1531–1555, 2004.

[32] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of Machine Learning Research*, vol. 3, pp. 1289–1305, 2003.

[33] G. Forman and I. Cohen, "Learning from little: Comparison of classifiers given little training," in *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2004, pp. 161–172.

[34] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, pp. 531–537, 1999.

[35] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[36] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[37] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," *Advances in Intelligent Computing*, pp. 878–887, 2005.

[38] T. Hertz, A. B. Hillel, and D. Weinshall, “Learning a kernel function for classification with small training samples,” in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 401–408.

[39] K. Huang, H. Yang, I. King, and M. Lyu, “Learning classifiers from imbalanced data based on biased minimax probability machine,” *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, no. 27, pp. II–558–II–563, 2004.

[40] J. V. Hulse, T. M. Khoshgoftaar, and A. Napolitano, “Experimental perspectives on learning from imbalanced data,” in *Proceedings of the 25th International Conference on Machine Learning*, 2007, pp. 935–942.

[41] N. Japkowicz, Ed., *Proceedings of the AAAI’2000 Workshop on Learning from Imbalanced Data Sets*, 2000, AAAI Tech Report WS-00-05.

[42] N. Japkowicz, “Supervised versus unsupervised binary learning by feedforward neural networks,” *Machine Learning*, vol. 42, no. 1/2, pp. 97–122, 2001.

[43] A. Jemal, M. J. Thun, L. A. G. Ries, H. L. Howe, H. K. Weir, M. M. Center, E. Ward, X.-C. Wu, C. Eheman, R. Anderson, U. A. Ajani, B. Kohler, , and B. K. Edwards, “Annual report to the nation on the status of cancer, 19752005, featuring trends in lung cancer, tobacco use, and tobacco control,” *Journal of the National Cancer Institute*, vol. 100, pp. 1672–1694, 2008.

[44] Y. jin Cui, S. Davis, C. kun Cheng, and X. Bai, “A study of sample size with neural network,” in *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, 2004, pp. 3444–3448.

[45] T. Jo and N. Japkowicz, “Class imbalances versus small disjuncts,” *SIGKDD Explorations*, vol. 6, no. 1, pp. 40–49, 2004.

[46] T. Joachims, “Training linear SVMs in linear time,” in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006, pp. 217–226.

[47] K. Kira and L. Rendell, “The feature selection problem: Traditional methods and new algorithm,” in *Proceedings of the 9th International Conference on Machine Learning*, 1992, pp. 249–256.

[48] R. Kohavi and G. John, “Wrappers for feature subset selection,” *Artificial Intelligence*, vol. 97, pp. 273–324, 1997.

[49] D. Koller and M. Sahami, “Toward optimal feature selection,” in *Proceedings of the 13th International Conference on Machine Learning*, 1996, pp. 284–292.

[50] I. Kononenko, “Estimating attributes: Analysis and extensions of RELIEF,” in *Proceedings of the 7th European Conference on Machine Learning*, 1994, pp. 171–182.

[51] M. Kubat and S. Matwin, “Addressing the curse of imbalanced data sets: One sided sampling,” in *Proceedings of the 14th International Conference on Machine Learning*, 1997, pp. 179–186.

[52] ——, “Learning when negative examples abound,” in *Proceedings of the 9th European Conference on Machine Learning ECML97*, 1997, pp. 146–153.

[53] J. Loughrey and P. Cunningham, “Overfitting in wrapper-based feature subset selection: The harder you try the worse it gets,” in *Proceedings of the 24th*

SGAI International Conference on Innovative Techniques and Applications of Artifcial Intelligence, 2004, pp. 33–43.

[54] O. Luaces, "MEX interface for SVMperf," http://web.me.com/ oluaces/, 2008.

[55] ——, "SVMperf MATLAB Spider object," http://web.me. com/oluaces/, 2008.

[56] O. Lund, M. Nielsen, C. Lundegaard, C. Kesmir, and S. Brunak, *Immunological Bioinformatics.* The MIT Press, 2005, pp. 99–101.

[57] P. Mahalanobis, "On the generalized distance in statistics," in *Proceedings of the National Institute of Sciences of India*, 1936, pp. 49–55.

[58] L. M. Manevitz and M. Yousef, "One-class SVMs for document classification," *Journal of Machine Learning Research*, vol. 2, pp. 139–154, 2001.

[59] H. Masnadi-Shirazi and N. Vasconcelos, "Asymmetric boosting," in *Procceedings of the 24th International Conference on Machine Learning*, 2007, pp. 609–619.

[60] L. Micó, F. Moreno-Seco, J. S. Sánchez, J. M. Sotoca, and R. A. Mollineda, "On the use of different classification rules in an editing task," in *In Joint IAPR International Workshops on Structural, Syntactic, and Statistical Pattern Recognition (SSPR/SPR06)*, 2006, pp. 747–754.

[61] T. Mitchell, *Machine Learning.* McGraw Hill, 1997.

[62] D. Mladenić and M. Grobelnik, "Feature selection for unbalanced class distribution and naive bayes," in *Proceedings of the 16th International Conference on Machine Learning*, 1999, pp. 258–267.

[63] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.

[64] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1226–1238, 2005.

[65] P. Pudil, J. Novovicova, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, pp. 1119–1125, 1994.

[66] A. Raskutti and A. Kowalczyk, "Extreme rebalancing for SVMs: a SVM study," *SIGKDD Explorations*, vol. 6, no. 1, pp. 60–69, 2004.

[67] M. Robert, R. C. Holte, L. E. Acker, and B. W. Porter, "Concept learning and the problem of small disjuncts," in *Proceedings of the 11th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1989, pp. 813–818.

[68] R. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, pp. 197–227, 1990.

[69] Y. Sun, M. Kamel, and Y. Wang, "Boosting for learning multiple classes with imbalanced class distribution," in *6th International Conference on Data Mining*, 2006, pp. 592–602.

[70] K. Ting, "The problem of small disjuncts: its remedy on decision trees," in *Proceedings of the 10th Canadian Conference on Artificial Intelligence*, 1994, pp. 91–97.

[71] G. Weiss, "Mining with rarity: A unifying framework," *SIGKDD Explorations*, vol. 6, no. 1, pp. 7–19, 2004.

[72] G. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *Journal of Artificial Intelligence Research*, vol. 19, pp. 315–354, 2003.

[73] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for support vector machines," in *Advances in Neural Information Processing Systems*, 2000.

[74] H. Xiong and X. Chen, "Kernel-based distance metric learning for microarray data classification," *BMC Bioinformatics*, vol. 7, p. 299, 2006.

[75] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *Journal of Machine Learning Research*, vol. 5, pp. 1205–1224, 2004.

[76] H. Zhang, "The optimality of naïve bayes," in *FLAIRS Conference*, 2004.

[77] Z. Zhao, J. Wang, H. Liu, J. Ye, and Y. Chang, "Identifying biologically relevant genes via multiple heterogeneous data sources," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 839–847.

[78] Z. Zheng, X. Wu, and R. Srihari, "Feature selection for text categorization on imbalanced data," *SIGKDD Explorations*, vol. 6, pp. 80–89, 2004.