11-3-2004

Applying the Wrapper Approach for Auto Discovery of Under-Sampling and Over-Sampling Percentages on Skewed Datasets

Ajay D. Joshi
University of South Florida

Applying the Wrapper Approach for Auto Discovery of Under-Sampling and Over-Sampling Percentages on Skewed Datasets

by

Ajay D. Joshi

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Lawrence O. Hall, Ph.D.
Rafael Perez, Ph.D.
Sudeep Sarkar, Ph.D.

Date of Approval:
November 3, 2004

Keywords: Machine Learning, Data Mining, SMOTE, RIPPER, imbalance, C4.5, F-value

ACKNOWLEDGEMENTS

I gratefully acknowledge Dr. Lawrence Hall for giving me the opportunity to work with him and guiding me through all the steps of this research work. I thank Dr. Nitesh Chawla for providing his valuable comments and suggestions. I am also grateful to Dr. Rafael Perez for providing me guidance and support during the course of my graduate studies at USF and for being on my committee. I thank Dr. Sudeep Sarkar for being on my committee and taking time and efforts to review my work. I am also indebted to my parents for their support and encouragement, and to my sister who gave me inspiration in every step.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

APPLYING THE WRAPPER APPROACH FOR AUTO DISCOVERY OF UNDER-SAMPLING & OVER-SAMPLING PERCENTAGES ON SKEWED DATASETS

Ajay D. Joshi

ABSTRACT

Machine learning applications are plagued by the imbalance observed among the class sizes in many real world datasets. A dataset is said to be skewed or imbalanced when its classes are very unequally represented. A naïve classifier learned from these skewed datasets is always biased towards the majority classes which constitute a major percentage of the samples in the dataset. As a result the accuracy on the minority classes is hampered. In many real world applications like network intrusion detection, cancer detection from mammography images, etc. the events of interest are very rare and the cost of not detecting these events is very high. Hence it very important to improve accuracies on the minority classes. It has been proposed previously that under-sampling of the majority classes can reduce the bias of the learned classifier and over-sampling of the minority classes - especially SMOTE (Synthetic Minority Over-sampling TEchnique) can boost the classifier accuracy on minority classes. But the question of how much under-sampling and over-sampling to be done for a particular induction learning algorithm and dataset remains. We present a wrapper approach for searching for the under-sampling and over-sampling (i.e. SMOTE) percentages for a particular learning algorithm for a given skewed dataset. We compare the results obtained by the classifiers built on wrapper selected under-sampled and SMOTEd datasets with the ones obtained by classifiers built on the original datasets to show a statistically significant improvement in accuracies over minority classes. This proves the efficacy of the wrapper approach in searching for the under-sampling and over-sampling percentages. Further, it provides an automated method to select the number of synthetic examples to be created.

CHAPTER 1

INTRODUCTION

The past decade has seen a remarkable increase in the amount of information being stored and distributed in electronic format over the internet [1]. With the increase in use of automated electronic data gathering and remote sensing devices, with the falling prices of data storage and computing power, data collection has become very cheap and has contributed to the explosion of accessible data. This has provided the Data Mining community with an exceptional opportunity for extracting nontrivial implicit information from massive datasets. In spite of this huge amount of real world data, only some of the data instances are of interest and usually they are in the minority. These data instances constitute the minority classes, while the remaining instances make up the majority classes. These kinds of datasets sets are called skewed or imbalanced datasets. Many machine learning datasets such as network intrusion detection, fraud transaction detection, medical diagnostics, etc. are hard to learn from without a significant bias towards the majority classes. In addition, many times the cost of misclassifying the interesting data instances in the minority classes as belonging to other majority classes is much higher than the cost of the reverse error.

For example, in network intrusion detection applications, where the illegal activity on the network is usually a very small percent of the normal activity, the aim of the application is to catch all network intrusions including the unseen network attacks by continuously monitoring for any unusual user activity and to keep a low false alarm rate. The issue of detecting future novel attacks has led to an escalating interest in data mining techniques for intrusion detection [2] [3] [4] [5] [6]. In the case of medical diagnosis, the cost of a false positive diagnosis generally results only in putting the patient through unnecessary medical tests whereas the result of a false negative diagnosis can be fatal. To make matters worse, normally the datasets on which medical diagnostic applications are built are also imbalanced as in the case of detecting the pixels in mammography images as cancerous or normal [7] [8] [9], where the

abnormal cancerous pixels are only a very small fraction of the total pixels. Previous studies related to fraud detection [10] have revealed an imbalance in the data of the order of 100 to 1. In some high energy physics applications [10] there is a 10,000 to 1 imbalance between the majority and minority classes. Some research in the domains of fraudulent telephone calls [11], telecommunication management [12], text classification [13] and detection of oil spills in satellite images [14] has been done to deal with the problem of imbalanced datasets. All these applications require a reasonably high detection rate for the minority classes and a minimal error rate for the majority classes.

The recent interest in the class imbalance problem has led to two special workshops held at AAAI in 2000 [15] and ICML in 2003 [16], and a special issue newsletter from SIGKDD Explorations [17] on learning from imbalanced datasets. Researchers in the machine learning community have dealt with the problem of class imbalance with various approaches like over-sampling the minority classes, under-sampling the majority classes, assigning different costs for different misclassification errors, learning by recognition as opposed to discrimination, etc. Lots of research has been done in comparing the various sampling methods and even a question of whether sampling is becoming the de facto standard for countering the imbalance in datasets has been raised in the special issue SIGKDD Explorations [17] editorial. With all this there is still no research on finding the approach to sampling required to get good classifier accuracies on minority classes. This provides us the motivation for our research.

The remainder of the thesis is organized into chapters as follows: Chapter 2 discusses some of the important previous work on the approaches used in alleviating the class imbalance problem and their limitations, the wrapper approach and its uses in different scenarios, how we intend to use wrappers and finally the performance metrics used in our study. Chapter 3 presents our wrapper Under-sample SMOTE algorithm used for finding the under-sampling and SMOTEing percentages for the skewed datasets. Chapter 4 provides a description of the datasets used in our study and presents experimental results which confirm the applicability of our approach. Chapter 5 discusses conclusions and proposes directions for future work.

CHAPTER 2

RELATED WORK

2.1. The Class Imbalance Problem and Solutions

As discussed in the Introduction chapter, the problem with the imbalanced class distribution observed in many real world data applications is significant and a lot of research has been done on this problem. The researchers have dealt with the problem of class imbalance using two main approaches. The first one is to assign distinct costs to training examples [15] [19] and the other is to re-sample the original dataset by either over-sampling or SMOTEing the minority classes and / or under-sampling the majority classes [7] [13] [20] [21] [22]. The details of these approaches and a few other important strategies are discussed in the following subsections.

2.1.1. Over-Sampling with Re-Sampling of Minority Classes

Over-sampling of minority classes can be done by re-sampling the examples from minority classes thus increasing the bias of the learned classifier towards them and increasing the accuracy on minority classes. Japkowicz [22] has experimented with over-sampling the minority classes in several artificial one dimensional datasets with varying concept complexities using multi-layered perceptrons and has found the approach effective in datasets with complex concepts. She has used two re-sampling methods, random re-sampling in which the minority class is re-sampled randomly until it consists of as many examples as the majority class and focused re-sampling in which only examples which occur on the boundary of minority and majority classes are re-sampled. Focused re-sampling was found to give no clear advantage over the random re-sampling. Similarly, Ling and Li [21] have used the over-sampling approach in the direct marketing domain. They have used ada-boosted Naïve Bayes and ada-boosted C4.5 with lift analysis as a measure of a classifier's performance on three real world datasets: a loan product promotion dataset and two datasets one from a major life insurance company and another from a company which ran "bonus

programs". They found that over-sampling of the minority class did not significantly improve the classifiers performance. There was some improvement for ada-boosted C4.5, but none for the ada-boosted Naïve Bayes classifier. This difference in performance for different learning algorithms shows that the amount of over-sampling done could also be a function of the type of inductive learning algorithm used to build the classifier in addition to the order of imbalance and the complexity of the dataset.

2.1.2. Under-Sampling of Majority Classes

Under-sampling the majority class can reduce the bias of the learned classifier towards it and thus improve the accuracy on the minority classes. Kubat and Matwin [20] have provided an intelligent solution for doing selective under-sampling of the majority class in a two class problem. The majority class examples are divided into four categories: safe, redundant, borderline and class-label noise examples. The redundant examples are eliminated using the 1-NN rule while the borderline and class-label noise examples are removed using the concept of Tomek links [23]. Thus this one-sided selection method removes examples from the majority class while leaving the minority class untouched. The evaluation of the trained classifiers was done using a geometric mean of accuracy on minority and majority classes. From the results, it was found that the under-sampling approach was effective, as there was a significant improvement in performance measures over most of the testbed datasets.

2.1.3. Assignment of Different Costs for Different Misclassification Error

The relative importance of different kinds of errors can be represented by a cost matrix. Suppose there are C classes, we have a C x C cost matrix, where the value in row i and column j gives the cost or loss of predicting a case to belong to class i, when it actually belongs in class j. Normally the cost is zero when i equals j and one when i is not equal to j, which gives us the normal classification error-rate. If the cost is one if i equals j and zero otherwise, then the cost measure is the common classification accuracy measure. Several studies in the literature have proposed approaches for finding optimal cost matrices which assign higher costs for misclassification of cases from minority classes and lower cost otherwise. One such study was done by Domingos, who proposed a method for making an arbitrary classifier cost-sensitive by wrapping a cost-minimizing procedure, called MetaCost [15], around it. A close connection between cost-sensitive approaches and sampling techniques has be found by [24][25].

2.1.4. Learning by Recognition

In this approach the classification of the examples is done using a recognition-based inductive scheme instead of the discrimination-based scheme where one of the two classes is mostly ignored i.e. examples from that class are not used in the training process. Japkowicz et al. [23] have proposed a Novelty Detection technique for concept learning which proceeds by recognizing the examples from one class rather than differentiating between the examples from two classes. The approach involves training an auto-encoder – a multi-layered perceptron to reconstruct the examples from one class at the output layer and then using the auto-encoder to recognize novel examples. The auto-encoder recreates the input with small error if it belongs to the class the auto-encoder is built upon or large error if it came from the other class. Japkowicz [22] built two auto-encoders, one on the majority class and the other on the minority class but failed to reach the performance obtained by either the random over-sampling or random under-sampling approaches. Tax [26] has done similar research using SVM which was found to be competitive [27] with other recognition based learners.

2.1.5. Discussion

Some studies [21] [22] have been done which combined under-sampling of majority classes with over-sampling by replication of minority classes. While Japkowicz [22] had found this approach very effective, Ling and Li [21] were not able to get significant improvement in their performance measures. Japkowicz [22] had experimented with only one-dimensional artificial data of varying complexity whereas Ling and Li [21] had used real data from a Direct Marketing problem. This might have been the reason for the discrepancy between their results. On the whole from the body of literature, it was found that under-sampling of majority classes was better than over-sampling with replication of minority classes and that combination of the two did not significantly improve the performance over under-sampling alone.

But later, Chawla et al. [7] introduced a new over-sampling approach for a two class problem that over-sampled the minority class by creating synthetic examples rather than replicating examples. They pointed out the limitation of the over-sampling with replication in terms of the decision regions in feature space for decision trees. They concurred that as the minority class was over-sampled by increasing amounts, for decision trees, the result was to identify similar but more specific regions in the feature space

as the decision regions for the minority class. This can be observed in the decision region plots for decision trees on the mammography dataset using 2 attributes as shown in Figure 2-1 [7] and Figure 2-2 [7].

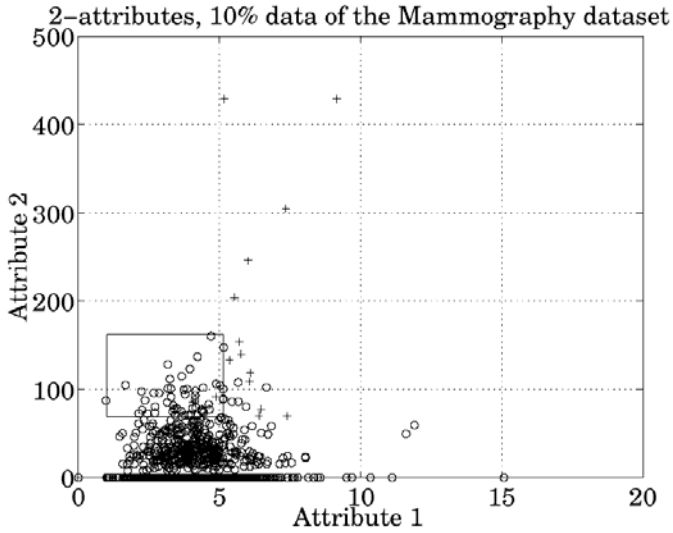

Figure 2-1. Decision Region for Majority Class Containing Three Minority Class Examples: Mammography Dataset

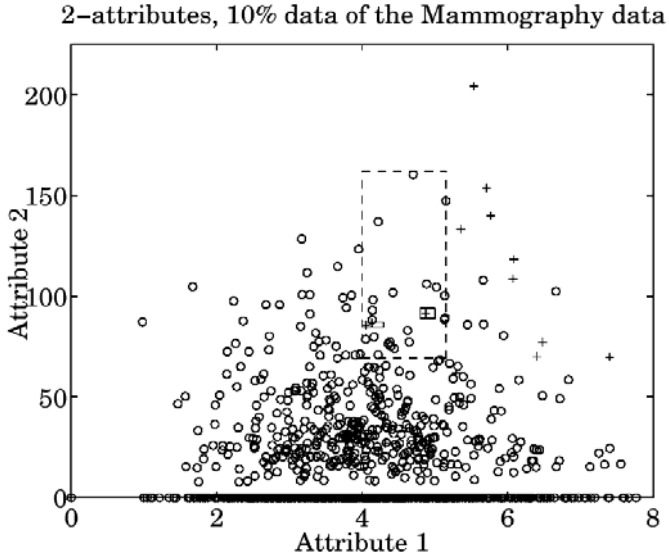

Figure 2-2. Very Specific Decision Regions for Three Minority Class Examples in Solid Line Rectangular Boxes: Mammography Dataset

From Figure 2-1 [7], in which the majority class samples are indicated by the 'o' marker and minority class samples by '+' marker, the region encapsulated by the solid line rectangle is the majority class decision region. It can be seen that this region includes three minority class samples i.e. the decision tree is misclassifying those minority class samples. But when the minority class is over-sampled using replication of the minority samples, it causes more splits in the decision tree leading to more terminal nodes/leaves as

the decision tree learning algorithm tries to fit more and more similar minority class data. This results in over-fitting the data and makes the decision regions for the minority class very specific. This is evident in Figure 2-2 [7], where it can be seen that the three previously misclassified minority class samples are classified correctly, but at the expense of very specific decision regions around them. Though one gets better accuracy for minority classes on training data, this does not hold true on the unseen test data. What we want is that the decision region for the minority class spreads into the majority class region as much as possible without much reduction, if any, in majority class accuracy; and thus be able to generalize well over unseen minority class data. The new approach called Synthetic Minority Over-sampling TEchnique (SMOTE) [7] overcomes the problem of specific decision regions by creating synthetic examples in the vicinity of the present minority class examples. A detailed description of this approach for over-sampling the minority classes, which is the approach used in our study, follows in the next section.

2.2. Synthetic Minority Over-sampling TEchnique (SMOTE)

The introduction of this approach of over-sampling for the minority class was inspired by a similar technique which was successful in handwritten character recognition [29]. The over-sampling was done by selecting "each minority class example and creating a synthetic example along the line segment joining the selected example and any/all of the *k* minority class nearest neighbors" [7, p.328]. In the calculations of the nearest neighbors for the minority class examples "a Euclidean distance for continuous features and the value Distance Metric (with the Euclidean assumption) for nominal features" [9, p.2] was used. For examples with continuous features, the synthetic examples are generated by taking the difference between the feature vectors of selected examples under consideration and their nearest neighbors. The difference between the feature vectors is multiplied by a random number between 0 and 1 and then added to the feature vector of the example under consideration to get a new synthetic example. For nominal valued features, a majority vote for the feature value is taken between the example under consideration and its k nearest neighbors [7]. This approach effectively selects a random point along the line segment between the two feature vectors. This strategy forces the decision regions of the minority class learned by the classifier to spread and effectively provides better generalization performance on unseen data. This can be seen in Figure 2-2 [7], where the region encapsulated by the dashed line rectangle is the decision region for the

minority class. This approach also reduces the size of the decision tree as the classifier does not learn specifics of the data by reducing over-fitting.

Chawla et al. [7] experimented with a combination of under-sampling of the majority class and SMOTE for the minority class, and found that the combination of both approaches performed better than under-sampling alone. However, the question of how much under-sampling and over-sampling to be done for a given dataset was not answered and our study tries to answer this very question.

The pseudo code[1] for generation of new synthetic minority class samples is shown in Figure 2-3 [9].

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

Figure 2-3. SMOTE Algorithm [9]

[1] SMOTE Perl script acquired from Nitesh Chawla (chawla@morden.csee.usf.edu)

```
Populate (N, i, nnarray) (* Function to generate the synthetic samples. *)
1.  while N ≠ 0 do
2.    choose a random number between 1 and k, call it nn. This step chooses one of the k
      nearest neighbors of i.
3.    for attr ← 1 to numattrs
4.      if attr = =  Continuous feature
5.        Compute: dif = Sample[nnarray[nn]][attr] – Sample[i][attr]
6.        Compute: gap = random number between 0 & 1
7.        Synthetic[newindex][attr] = Sample[i][attr] + gap * dif
8.      else
9.        attr_value = majority vote for the attr value from k nearest neighbors. If no majority
          then choose at random.
10.       Synthetic[newindex][attr] = attr_value
11.     end if
12.   end for
13.   newindex++
14.   N = N – 1
15. end while
16. return (* End of Populate *)
    End of Pseudo-Code
```

Figure 2-3. SMOTE Algorithm Contd. [9]

2.3. The Wrapper Approach

Kohavi et al. [30] were the first to introduce the wrapper approach to the mainstream Data Mining community. They successfully used the wrapper approach to search for an optimal feature subset customized to a specific induction learning algorithm and domain. The idea behind the wrapper approach is very simple and is shown in Figure 2-4, where the induction learning algorithm which is used as a black box is run repeatedly on a distinct portion of the dataset using various feature subsets. Some performance measure is used to evaluate the classifier built on each feature subset using a set aside distinct portion of the dataset, and the feature subset with the highest evaluation is used as the final set to build the final classifier on all the data instances in the training set. The resulting classifier can then be evaluated on an independent test set that is not used during the search process to assess the efficacy of the wrapper approach in selecting the feature subset.

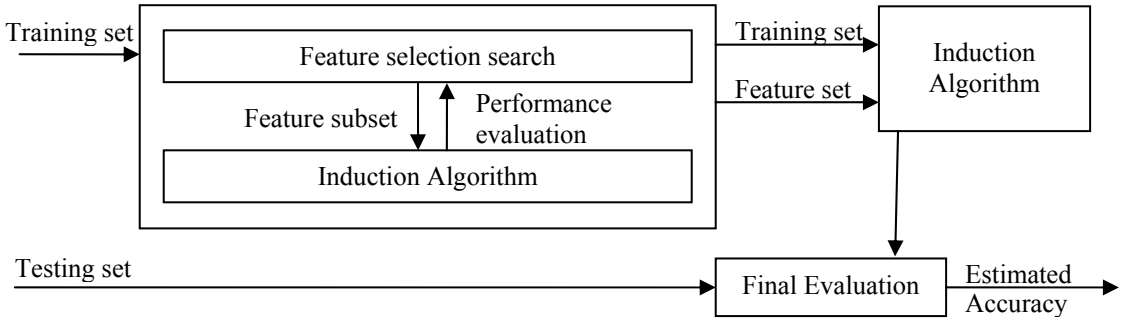

Figure 2-4. The Wrapper Approach to Feature Subset Selection

After Kohavi et al. successfully used the wrapper approach in the feature subset selection problem, there were many researchers who experimented with the wrapper approach in various contexts. Langley and Sage [31] used the wrapper approach in selecting the features for a Naïve-Bayes classifier. Pazzani [32] created super-features by combining the base features for a Naïve-Bayes classifier by using the wrapper approach and demonstrated that it really was able to find the correct combination of features when they interacted. A significant improvement over the original K2 algorithm was shown by Singh and Provan [33]

when they selected the features for Bayesian networks using the wrapper approach. Again, Kohavi and John [34] demonstrated the use of the wrapper with other search methods using probabilistic estimates for feature subset selection.

Other than feature selection, the wrapper approach has been used for many other problems. Kohavi and John [35] applied the wrapper approach for tweaking the parameters of C4.5 for maximal performance. The wrapper approach was used by Skalak [36] in an interesting fashion to select the training instances (prototype subset) instead of the features in connection with nearest-neighbor classifiers.

Our algorithm, which searches for under-sampling and over-sampling percentages for minority and majority classes respectively, includes a modified version of the wrapper algorithm.

2.4. Measurements and Metrics

Traditionally, the performance evaluation of machine learning algorithms was done using just the predictive classification accuracy, which did not capture the essence of recognizing the minority classes in skewed datasets with diverse costs of errors. In skewed datasets, where the majority class constitutes about 98% - 99% of all the data instances, a trivial classifier which predicts everything as belonging to the majority class can achieve an accuracy of 98% - 99% which appears great on the surface. So it is apparent that for domains with imbalanced distributions, classification accuracy is not an appropriate performance measure. Previous studies involving imbalanced datasets have used various different performance measures suitable to this type of problem. For example, Kubat and Matwin [15] used the geometric mean of accuracies measured on the minority and majority class with the idea to improve the accuracy on both classes while keeping them balanced. But this metric did not take into account the different costs for different misclassification errors. Several studies [7] [37] [38] [39] have used ROC (Receiver Operating Characteristics) analysis as a standard technique for summarizing the classifiers performance over a range of tradeoffs between true positives and false negatives and AUC (Area Under the Curve) as the performance metric for ROC curves. In [7], for generation of ROC curves, the data set was SMOTEd for minority classes at a specific percentage while the under-sampling of the majority class was done over a range of percentages ranging from 0% to 100% to get the points on the ROC curve. Then the SMOTE percentage, which was used to SMOTE the dataset for the minority class which yielded the best AUC

measure, was selected as the best SMOTE percentage. Thus the AUC measure gives the best SMOTE percentage for an under-sampled percentage of the majority class ranging from 0% to 100%. But what we are really interested in, is the specific amount of SMOTE and under-sampling percentages i.e. a single point on the ROC curve that will give us the best tradeoff between the true positives and false positives. The AUC measure does not provide that, and so is not considered to be an appropriate performance measure here. Also the AUC measure does not take into account the cost of classifying a positive case as negative (FN) and also is very hard to extend for multiple class problems.

The F-value [40] [41] measure which takes into consideration the false positives and false negatives along with true positives was the right metric for our study. A confusion Matrix as shown in Table 2-1 is used in calculation of the F-value. In our case the minority class under consideration is deemed as the positive class, whereas all other classes together constitute the negative class.

Table 2-1. Confusion Matrix Defining Four Possible Classification Scenarios

	Predicted Positive Class	Predicted Negative Class
Actual Positive class	True Positives (TP)	False Negatives (FN)
Actual Negative Class	False Positives (FP)	True Negatives (TN)

The F-value metric incorporates two other measures: Precision which gives us the measure of correctness of the classifier in predicting the actual positive or minority class, whereas Recall gives us the measure of the percentage of positive or minority class examples predicted correctly. Using the Table 2-1, Precision, Recall and F-value are calculated as follows:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \tag{2.1}$$

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{2.2}$$

$$\text{F - value} = \frac{(1+\beta^2) \times \text{Recall} \times \text{Precision}}{\beta^2 \times \text{Recall} \times \text{Precision}} \tag{2.3}$$

Where β controls the relative importance assigned to recall and precision.

For any machine learning algorithm, it is desirable to improve the recall without sacrificing the precision. However, the optimal independent parameter settings to optimize for Recall or Precision are

often conflicting and optimizing both of them simultaneously may be difficult. Both of these measures are included by the F-value and so the goodness of the classifier can be measured by the F-value.

2.5. Discussion

From the previous research on boosting accuracy for minority classes, a combination of over-sampling of minority classes using SMOTE and under-sampling of majority classes has been found effective. However, a method to find how much to SMOTE and under-sample is still not known. The wrapper approach has been successfully used in many domains viz. searching for the optimal feature subset, tuning parameters of induction learning algorithms, etc. So in this study we have used the wrapper approach in searching for SMOTE percentages for minority classes and under-sampling percentages for majority classes, guided by the F-value as a performance metric, for a specific induction learning algorithm on imbalanced datasets. The next chapter presents the Wrapper Under-sample SMOTE algorithm.

CHAPTER 3

WRAPPER UNDER-SAMPLE SMOTE ALGORITHM

As indicated in the previous chapter, our Wrapper Under-sample SMOTE algorithm uses the wrapper approach in searching for the best under-sampling and SMOTE percentages. From previous work, under-sampling was found to be better than over-sampling with replication. Testing for each pair of under-sampling and SMOTE percentages will result in a search space of intractable size, and hence the information from previous studies is used as a heuristic for the search where searching for under-sampling percentages is done first then followed by a search for the SMOTE percentages. This strategy will first remove the "excess" majority class examples which will result in a reduction in the size of the training dataset and thus reduce the learning time required to build the classifier. Then over-sampling of the minority class examples will add synthetic minority class examples and increase the generalization performance of the classifier over the minority classes. Figure 3-1 shows our Wrapper Under-sample SMOTE algorithm [2] which can handle multiple majority and minority classes.

We have also experimented with a Brute Force Search method which varies both the under-sampling and the SMOTE percentage simultaneously over a discrete search space and covers all discrete valued combinations of under-sampling and SMOTE percentage pairs. Due to its time consuming nature we have tested this strategy on only two datasets. The details about this technique and comparison of results to our algorithm are presented in a later section of Chapter 4.

[2] Perl scripts and datasets can be acquired at http://morden.csee.usf.edu/wrapsmote/

Algorithm *WRAPPER_UNDERSAMPE_SMOTE* (*MinorList, MajorList, NoFolds*)

Input: List of minority classes *MinorList*
List of majority classes *MajorList*
Number of cross-validation folds *NoFolds*

Output: Wrapper selected UnderSampling percentages for majority classes and
Wrapper selected SMOTE percentages for minority classes

1. Set the under-sampling percentage for the majority class to 100% (No under-sampling) in *UnderSampleList* and the smoting percentage for the minority class to 0 (No Smote) in *SmoteList*
2. *Do NoFolds* cross-validation on the training data without under-sampling and smote and get baseline *MinorFvalues* & *MajorFvalues* and assign them to *BestMinorFvalues* and *BestMajorFvalues*
3. **if** *MajorList* is not empty
4. **then** *WRAPPER_UNDERSAMPLE*(*UnderSampleList*)
5. **end if**
6. **if** *MinorList* is not empty
7. **then** *WRAPPER_SMOTE*(*SmoteList*)
8. **end if**

Figure 3-1. Wrapper Under-sample SMOTE Algorithm

3.1. Wrapper Based Algorithm to Select Under-sampling Percentages

This algorithm implements a wrapper which performs the search through the parameter space of under-sampling percentages for the majority class(es), using the chosen induction learning algorithm as a part of the evaluation function for a five-fold cross-validation over the training data. The purpose of the wrapper is to search for the state in the parameter space with the highest evaluation score guided by some heuristic function. Since the actual performance of the classifier will be assessed on the training data only, the estimated performance over a five fold cross-validation is utilized in the heuristic function in guiding the search process.

The wrapper starts with no under sampling for all majority classes and obtains baseline results on the training data. Then in a step-by-step greedy fashion it traverses through the search space of under-sampling percentages to seek better performance over the minority classes. The search process continues as long as it

does not hamper the accuracy of the minority classes (normally the accuracy of the minority classes increases) or drop the accuracy over majority classes more than some specified amount (generally 5%).

Table 3-1. Hypothetical Scenario 1 – Performance Metrics

Baseline	Actual Minority Class	Actual Majority Class
Predicted Minority class	119	51
Predicted Majority Class	115	9779
F-value minority class		0.589
F-value majority class		0.992

After under-sampling	Actual Minority Class	Actual Majority Class
Predicted Minority class	131	66
Predicted Majority Class	103	9764
F-value minority class		0.608
F-value majority class		0.991

Table 3-2. Hypothetical Scenario 2 – Performance Metrics

Baseline	Actual Minority Class	Actual Majority Class
Predicted Minority class	119	51
Predicted Majority Class	115	9779
F-value minority class		0.589
F-value majority class		0.992

After under-sampling	Actual Minority Class	Actual Majority Class
Predicted Minority class	122	66
Predicted Majority Class	112	9764
F-value minority class		0.578
F-value majority class		0.991

For example, from the hypothetical scenario 1 as shown in Table 3-1, one can see that after under-sampling the majority class, the F-value on the minority class is increased by 1.9% points from 0.589 baseline F-value at an expense of only 0.1% points drop in the majority class F-value. In cases like this, the wrapper algorithm continues to under-sample the majority class at higher under-sampling percentages until the average F-value over the majority classes falls below some specified amount. But sometimes as in scenario 2 in Table 3-2, when under-sampling of the majority class is performed, the F-value on the minority class can be reduced due to a high reduction in precision versus a small increase in recall. So to catch both these conditions in multiple minority/majority class problems we use average F-value of minority classes and majority classes in our heuristic function. After one of the conditions is met, the wrapper search for under-sampling is terminated and the search for SMOTE percentage for minority classes begins. The details of this algorithm are presented in Figure 3-2.

```
WRAPPER_UNDERSAMPLE(UnderSampleList) (* The Function implements a wrapper to
search for the best under-sampling percentage for majority classes. *)
1.  foreach MClass in MajorList (*MajorList contains one or more majority  classes *)
2.     StopUnderSamplingFlag[MClass] = false
3.  end foreach
4.  do
5.     foreach MClass in MajorList
6.        if StopUnderSamplingFlag[MClass] = false
7.           UnderSampleList[MClass] = UnderSampleList[MClass] – x; (* generally 10% *)
8.           for Fold ← 1 to NoFolds (* NoFolds cross-validation runs *)
9.              OutputTrainingData(Fold, UnderSampleList)
10.             OutputTestingData(Fold)
11.             Build classifier on the under-sampled training set and evaluate on validation set
12.          end for
13.          Update MinorFvalues & MajorFvalues
14.          if Average(MinorFvalues) < Average(BestMinorFvalues)
15.             UnderSampleList[MClass] = UnderSampleList[MClass] + x; (* reset to previous*)
16.             StopUnderSamplingFlag[MClass] = true
17.          else if Average(MajorFvalues) < Average(BestMajorFvalues) * LossFactor (* 5% *)
18.             UnderSampleList[MClass] = UnderSampleList[MClass] + x; (* reset to previous*)
19.             StopUnderSamplingFlag[MClass] = true
20.          else
21.             Update BestMinorFvalues
22.          end if
23.       end if
24.    end for
25.    while (StopUnderSamplingFlag for atleast one MClass = false)  (* Exit the loop when
       StopUnderSamplingFlag for all majority classes is set to true *)
26. return (* End of WRAPPER_UNDERSAMPLE *)
```

Figure 3-2. Wrapper Under-sample Algorithm

3.2. Wrapper Based Algorithm to Select SMOTE Percentages

This algorithm is very similar to the Wrapper Under-sample algorithm. The under-sampling percentages for the majority classes are now fixed to those found so far by the Wrapper Under-sample algorithm and the corresponding best F-value results for minority classes are baselined. Now the Wrapper SMOTE algorithm continues to step through the search space for the SMOTE percentage of minority classes in a greedy fashion and obtains new performance estimates using the five-fold cross-validation over the training data. If the average F-value over the minority classes is improved, then the current results are again baselined, and the search continues to find even better SMOTE percentage combinations for the minority classes. In some scenarios, even if the average F-value for the minority classes over five-fold cross-validation is reduced it can be attributed to the randomness of SMOTEing. Also assuming that more SMOTE will add more information and will give better accuracies over minority classes (though not always) true, we look ahead in the parameter search space by SMOTEing minority classes at higher SMOTE percentages. This sometimes results in a better true positive rate at an increased false positive rate which is normally acceptable in domains with imbalanced datasets. The search for the SMOTE percentage for a minority class stops when the average F-value for the minority classes cannot be improved even after two look aheads at a higher SMOTE percentage are performed. The pseudo code for the wrapper SMOTE algorithm is presented in Figure 3-3.

One may note that we do not perform a look ahead for under-sampling, as we believe that under-sampling effectively reduces the information contained in the data and thereby reduces accuracy by reducing the coverage of the built classifier. The sole purpose of under-sampling is to reduce the redundancy of majority class examples without much reduction in the accuracy over all classes.

```
WRAPPER_SMOTE(SmoteList) (* The Function implements a wrapper to search for best
SMOTE percentage for minority classes. *)
1.  foreach MClass in MinorList  (*MinorList  contains one or more minority classes *)
2.     StopSmoteFlag[MClass] = false
3.     LookupAhead[MClass] = 1
4.  end foreach
5.  do
6.     foreach MClass in MinorList
7.        if StopSmoteFlag[MClass] = false
8.           SmoteList[MClass] = SmoteList [MClass] +  LookupAhead[MClass] x y;
             (* y generally 100% *)
9.           for Fold ← 1 to NoFolds (* NoFolds cross-validation runs *)
10.             OutputTrainingData(Fold, UnderSampleList , SmoteList)
11.             OutputTestingData(Fold)
12.             Build classifier on under-sampled and SMOTEd training set and evaluate on
                validation set
13.          end for
14.          Update MinorFvalues & MajorFvalues
15.          if Average(MinorFvalues) < Average(BestMinorFvalues)
16.             SmoteList[MClass] = SmoteList [MClass] - y; (* reset to previous SMOTE*)
17.             if LookupAhead[MClass] < LookupAheadValue (* generally 3 *)
18.                LookupAhead[MClass] = LookupAhead[MClass] + 1
19.             else
20.                StopSmoteFlag[MClass] = true
21.          else
22.            Update BestMinorFvalues
23.          end if
24.       end if
25.    end for
26. while (StopSmoteFlag for atleast one MClass = false)   (* Exit the loop when
    StopSmoteFlag for all minority classes is set to true *)
27. return (* End of WRAPPER_UNDERSAMPLE *)
    End of Pseudo-Code
```

Figure 3-3. Wrapper SMOTE Algorithm

3.3. Cross-Validation of System

We have presented an approach for searching under-sampling and SMOTE percentages for imbalanced datasets, which when used to build classifiers will result in improved accuracy on minority classes over unseen data. Since the algorithm does not have access to unseen data, it uses five-fold cross-validation over training data as an evaluation function in forecasting the estimated accuracy over unseen data. These accuracy estimates will hold true only when either the training data is a good representative of the actual data distribution or the wrapper strategy does not over-fit the training data. If the training data is not a good representative of the actual data, no strategy can help. So the only thing which remains is to see whether the wrapper approach finds under-sampling and SMOTE levels which when used to build a classifier, do not over-fit the training data.

To ascertain that the wrapper approach is really able to get good under-sampling and SMOTE percentages, we do a ten-fold cross-validation in which the original dataset is stratified into ten disjoint sets from which ten distinct testing sets and ten training sets are created. The wrapper Under-sample SMOTE algorithm which uses five-fold cross-validation of the training set finds the under-sampling and SMOTE percentages for a particular training fold dataset (one of the ten folds for cross-validating the system). Then the whole training set is under-sampled and SMOTEd with wrapper selected percentages, a classifier is built on the updated training data and evaluated on the test data unseen during the wrapper process. Due to the inherent random nature of under-sampling and SMOTEing, to get a fair estimate of the performance obtained on the testing set, the process of training and testing with wrapper selected under-sampling and SMOTE percentages is done for a total of five times to get an averaged (more stable) accuracy measure.

The wrapper Under-sample SMOTE algorithm is run on total ten training sets to obtain a total of ten pairs of under-sampling and SMOTE percentages. These ten pairs of wrapper selected under-sampling and SMOTE percentages are evaluated on corresponding ten testing sets as explained above to cross-validate the whole system. To summarize, on each of the 10 folds, training and testing for wrapper selected SMOTE and under-sampling percentage is done five times i.e. SMOTE and under-sampling is done for a total of 50 times for cross-validation to get average stable results

The next chapter describes the experimental setup, the datasets, the two inductive learning algorithms used in our research, the tests used to affirm the statistical significance of the results obtained by wrapper approach and the results.

CHAPTER 4

EXPERIMENTS

4.1. Datasets

There were seven real datasets used as a testbed in our research –

- *The Phoneme Dataset,*
- *The Satimage Dataset,*
- *The Mammography Dataset,*
- *The Forest Cover Dataset,*
- *The Pima Indian Diabetes Dataset,*
- *The Oil Dataset and*
- *The KDD-cup 99: Network Intrusion Detection Dataset (Two versions).*

A brief summary of the following datasets is presented in Table 4-1 and further details are given in later subsections.

Table 4-1. Summary of Datasets

No.	Dataset	# of Examples	# classes	# of Majority class examples		# of Minority class examples		# of attributes	# of continuous attributes
1	Phoneme	5404	2	3818		1586		5	5
2	Satimage	6435	2	5809		626		36	36
3	Mammography	11183	2	10923		260		6	6
4	Forest cover	38501	2	35754		2747		54	54
5	Pima Indian	768	2	500		268		8	8
6	Oil	937	2	896		41		49	49
7	Modified KDD-cup 99: Network Intrusion Detection	69980	5	Normal	35000	U2R	267	41	34
				Dos	25988	R2L	3912		
				Probe	4813				

4.2. Machine Learning Algorithms

It is our hypothesis that, the under-sampling and SMOTE levels required to obtain, for a particular dataset, the best performance are not constant and change depending upon the machine learning algorithm used in building the classifier. So to evaluate this hypothesis we have used two different induction learning algorithms: C4.5 [42] and RIPPER [47]. The next two sub-sections discuss upon these two learning algorithms.

4.2.1. C4.5

The C4.5 [42] algorithm belongs to the family of decision tree induction learning algorithms, and is a much improved version of the ID3 [48] algorithm. In decision trees, the classification of a particular pattern starts at the root node, where a test on the values of a particular attribute of the example pattern is done. The node is split into branches depending on the different values (or sets of values) taken by the test attribute. Based on the value of the attribute, an appropriate branch to a subsequent or child node is taken. Next, the chosen test at the child node under consideration, which can be considered as the root of the following sub-tree, is performed and the test pattern is sent down the appropriate path. This procedure continues iteratively until a leaf node is reached, which has no further test. Each leaf node bears a classification label and a test pattern is assigned the classification of the leaf node reached. It is evident that each path from root to leaf is a conjunction of the attribute tests, and the tree is a disjunction of conjunctions of the attribute tests along all paths in the tree. Sometimes, decision tree learners over-fit the training data by unnecessarily splitting the leaf nodes of an ideal decision tree. To reduce the effect of over-fitting, the C4.5 algorithm has a provision for pruning the resulting decision tree using heuristics based on the statistical significance of splits where each sub-tree is recursively replaced with a following leaf node if the projected error rate for the leaf node is less than that for the sub-tree. A modified version of C4.5 release 8 called USFC4.5 from the University of South Florida was used with default parameters for all the experiments. USFC4.5 with default parameters settings produces identical output to C4.5 release 8. The added functionality in USFC4.5 was used to interface the learner with our wrapper under-sample SMOTE script.

4.2.2. RIPPER

The RIPPER [47] algorithm is an induction learning technique belonging to a family of propositional rule learning systems. RIPPER which stands for 'Repeated Incremental Pruning to Produce Error Reduction' mainly consists of an improved version of Incremental Reduced Error Pruning [49] (IREP) called IREP* [47] with some passes through the initial rules which do rule optimization. IREP* integrates reduced error pruning with a separate-and-conquer rule learning algorithm. The separate-and-conquer rule learning algorithm [50] builds rules in a greedy fashion, one rule at a time. Then once a rule is found, all examples covered by that rule are removed. In order to build a rule, all uncovered examples are partitioned into a growing set (two-thirds of examples in this case) and pruning set, and a rule is built using a propositional version of FOIL [51] which tries to optimize for information gain. Then after growing a rule, it is immediately pruned using the pruning set. Adding rules to the rule-set is stopped when a rule is learned that has an error rate greater than 50%. IREP* incorporates a metric to guide rule pruning and an MDL-based heuristic [52] for determining how many rules should be learned. After learning a rule-set which covers all the data examples except the default class examples, n iterations of the optimization phase which mimic the effect of non-incremental reduced error pruning are done. The optimization phase consists of building two alternative rules for each rule in the rule-set: a replacement rule built using the growing and pruning set to reduce the error rate of the entire rule-set and a revision rule which is a revised version of the rule under consideration also optimized to reduce the error rate of the entire rule-set, and then selecting one of them using the MDL heuristic. Once the rule-set is modified by the optimization phase, IREP* is again executed to build more rules on the examples which are not covered by this optimized rule-set. It was found that RIPPER with two optimization passes, called RIPPER2, was extremely competitive with C4.5rules [42]. RIPPER2 is used in our experiments. All other parameters are left at their default values.

4.3. Consistency of Training, Testing and Validations Sets

Four experimental runs were done on each of the seven datasets using USFC4.5 and RIPPER as induction learning algorithms and the wrapper script was executed to search for under-sampling and SMOTE percentages together and also the SMOTE percentage with no under-sampling. As described earlier in the

Cross-Validation subsection, the wrapper script was executed 10 times on 10-fold cross-validation sets, which again repeatedly does a five-fold cross-validation of training data to find under-sampling and SMOTE percentages. In order to eliminate the effects that may arise from the creation of random training, testing and validation sets across the four experimental runs, the same seed was used for initialization of the random number generator so that the data samples selected across the four experimental runs are identical. Likewise, the order in which the data samples were presented to the learning algorithms was also maintained so as to eliminate its effects on order sensitive learners like RIPPPER which internally partitions the training data into a growing and pruning set. For example, if RIPPER picks examples 1, 3, 9,….etc. as a growing set, then if the data samples are provided in a different order, RIPPER still picks examples 1, 3, 9,….etc. thus changing the growing set and resulting in a different learned rule set. This is not desired and so it was necessary to maintain the order of examples presented to the RIPPER learner.

4.4. Tests for Statistical Significance

Tests for Statistical Significance are very important as they provide information about whether observed differences in results are really meaningful or simply a chance happening. It is not possible to use these tests on individual results, but when groups of results are available, we can use statistical tests to affirm which results are statistically better than others. So to establish, that the wrapper approach is successful in selecting the under-sampling level for majority classes and the SMOTE level for minority classes, we did a 10-fold cross-validation on all datasets and compare wrapper results for statistical significance over baseline results using three performance metrics – minority class TP-rate and F-value and majority class F-value.

The wrapper results were obtained when the classifier was built on the data whose baseline distribution had been changed using under-sampling and SMOTE. This change in distribution was part of the experiment, and the changes in the wrapper results were due to the altered distribution of training data and should not be attributed to random effects. With this we can say that the baseline and wrapper results obtained on a specific cross-validation run were dependant or paired.

The t-test which is the most commonly used method to evaluate the differences in means between two groups has two variants: the unpaired t-test for independent groups and the paired t-test for paired groups of

data. In our case, since the groups of 10 baseline and wrapper results that are to be compared are based on the same base datasets which are tested twice (before and after under-sampling and SMOTE), a considerable part of the within-group variation in both groups of results can be attributed to the initial individual differences between the 10 fold datasets. The paired t-test uses the information regarding the pairing of the data from two groups to identify and exclude an important source of within-group variation (or error) and effectively increases the test's sensitivity and produces more realistic results when compared with the un-paired t-test. The t-statistic is basically the ratio of mean difference between two paired groups and the standard deviation of differences (standard error) between paired values of two groups, and is calculated as follows:

$$\text{Mean}_{\text{difference}} = \frac{\sum_{1}^{n} x_i - y_i}{n} \tag{4.1}$$

$$\text{StdDev}_{\text{difference}} = \sqrt{\frac{\sum_{i=1}^{n} ((x_i - y_i) - \text{Mean}_{\text{difference}})^2}{n-1}} \tag{4.2}$$

$$t = \frac{\text{Mean}_{\text{difference}}}{\text{StdDev}_{\text{difference}} / \sqrt{(n)}} \tag{4.3}$$

Where x_i is from group A, y_i is from group B and n is the number of paired observations.

Thus the paired t-test compares the magnitude of the difference between paired group results to the variation among the differences and if the mean difference is large compared with standard error, then the results are statistically different. One underlying assumption when using the t-test is that variables should be normally distributed within each group. If we assume that the datasets we are using to build the classifiers have normally distributed features which is generally true, then we can extend this assumption to the cross-validation sample datasets drawn from the original dataset. Continuing, we can also assume that the baseline and wrapper results obtained from the *n* cross-validation folds are normally distributed. Hence we can safely use the paired t-test for comparing baseline and wrapper results. Also in [53] it was stated that the t-test still works well even if the assumption of normality is slightly violated. Since the direction of increase or decrease in the performance measures is known, we use a one-tailed analysis with $t_{critical} = 1.833$ at level of significance $\alpha = 0.05$ and degree of freedom df = 9 (one less than the number of paired observations).

Though statistical tests are important in assessing the results, one should still distinguish between statistical and practical tests as suggested by Keen [54]. Statistical tests are only concerned about establishing with some degree of confidence that one group of results are better or worse than the other. Whereas practical tests are concerned with the magnitude of the difference noticeable by the user between the two sets of results and are also important. They complement the statistical tests. So along with the t-statistic we also provide the percentage increase observed in the TP-rate and F-value on the minority class and percentage decrease observed in the F-value on the majority class.

4.5. Results

4.5.1. Phoneme Dataset

The Phoneme dataset was obtained from the ELENA project [44]. The purpose of the dataset is to distinguish between nasal (class 0) and oral sounds (class 1). There are five features which are all numeric. The classes are skewed with 29.35% of the total examples belonging to the minority class. There are 3,818 examples in class 0 and 1,586 examples in class 1. The results obtained from four experimental runs: C4.5 with 'SMOTE only' and 'under-sampling with SMOTE' and RIPPER with 'SMOTE only' and 'under-sampling with SMOTE' are shown in Table 4-2.

As we expected to get an increase in the TP-rate and the F-value for the minority class the table lists the 'percentage increase' from the baseline TP-rate and F-value respectively for the minority class. Whereas for the majority class, the 'percentage decrease' from the baseline F-value is calculated as under-sampling the majority class and SMOTEing minority class reduces the bias of the classifier towards the majority class thus reducing accuracy over it. The baseline results refer to the results obtained on the original dataset which is not under-sampled and SMOTEd. The symbol '√ ' indicates that wrapper results are statistically significantly better than baseline results, '×' indicates that wrapper results are statistically significantly worse than baseline results and '-' indicates that there is no statistical difference between the two results. The negative sign before the t-statistic indicates that the wrapper results are better than the baseline results. If the absolute t-statistic value is greater than $t_{critical}$ = 1.833, we can say at the 95% confidence level that the two results groups are statistically significantly different.

Table 4-2. Results for Phoneme Dataset

	Algorithm	C4.5		RIPPER	
		SMOTE only	Under-sampling & SMOTE	SMOTE only	Under-sampling & SMOTE
	Average SMOTE percentage	160%	150%	200%	150%
	Average Under-sampling percentage	100%	89%	100%	90%
Average Minority class TP-rate	Baseline	0.783	0.783	0.725	0.725
	Wrapper	0.866	0.875	0.879	0.876
	% increase	9.56%	10.51%	17.47%	17.21%
	t-stat	-8.333	-8.567	-8.375	-11.145
	Significance	√	√	√	√
Average Minority class F-value	Baseline	0.773	0.773	0.748	0.748
	Wrapper	0.787	0.780	0.761	0.758
	% increase	1.77%	0.86%	1.62%	1.33%
	t-stat	-2.828	-0.949	-2.057	-2.229
	significance	√	-	√	√
Average Majority class F-value	Baseline	0.904	0.904	0.9	0.9
	Wrapper	0.898	0.891	0.877	0.876
	% decrease	0.65%	1.38%	2.59%	2.68%
	t-stat	3.218	3.971	3.997	5.752
	significance	×	×	×	×

Detailed Results for each fold are provided in the charts in Figure 4-1 to Figure 4-4, where the x-axis is labeled using the percentage of SMOTE and under-sampling performed to obtain the results and the fold number. As indicated earlier, due to the inherent randomness in under-sampling and SMOTEing, to get fair performance measures over the test set, the classifiers were built and evaluated five times using wrapper selected under-sampling and SMOTE percentages. The standard deviation from the average TP-rates and F-values is shown in the charts using standard deviation bars for wrapper results. The baseline results are always constant and do not have standard deviation bars.

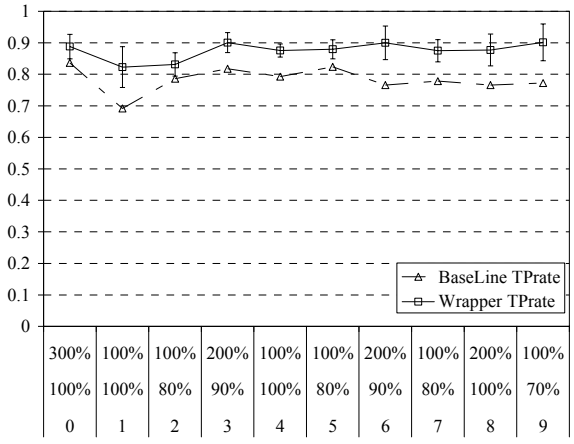

Figure 4-1. True Positive Rate for Phoneme Data Using C4.5 – SMOTE with Under-sampling

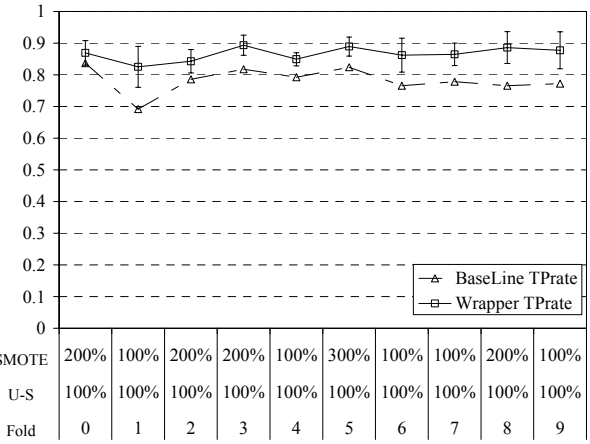

Figure 4-2. True Positive Rate for Phoneme Data Using C4.5 – SMOTE Only

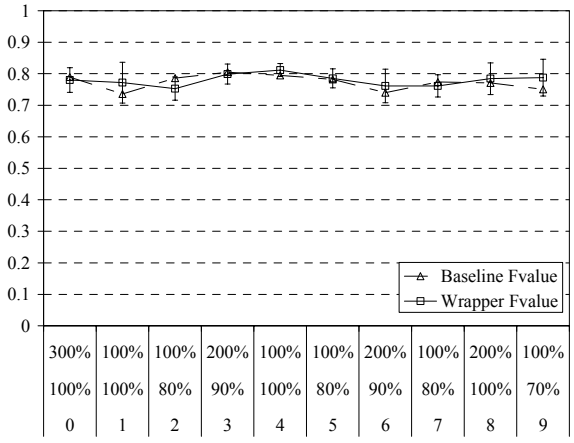

Figure 4-3. F-value for Phoneme Data Using C4.5 – SMOTE with Under-sampling

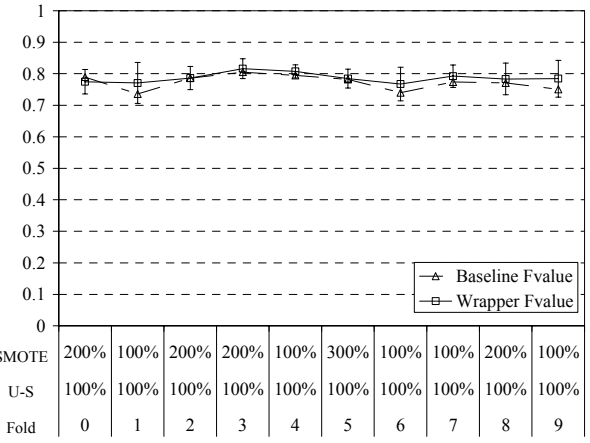

Figure 4-4. F-value for Phoneme Data Using C4.5 – SMOTE Only

From Figure 4-1 and Figure 4-2, it can be seen that the wrapper approach for C4.5 was able to significantly improve TP-rates for the minority class (9.56% and 10.51% increase on average for 'SMOTE only' and 'under-sampling & SMOTE' respectively) on all 10 test sets, with the wrapper TP-rate using 'under-sampling & SMOTE' better than wrapper TP-rate using 'SMOTE only'. In the case of the 'SMOTE only' scenario the wrapper F-values for the minority class were statistically better than the baseline F-values as seen in Table 4-2. Also the F-values on the majority class were statistically worse than the baseline F-values, but the amount of decrease was very small (0.65% and 1.38% on average for 'SMOTE only' and 'under-sampling & SMOTE' respectively).

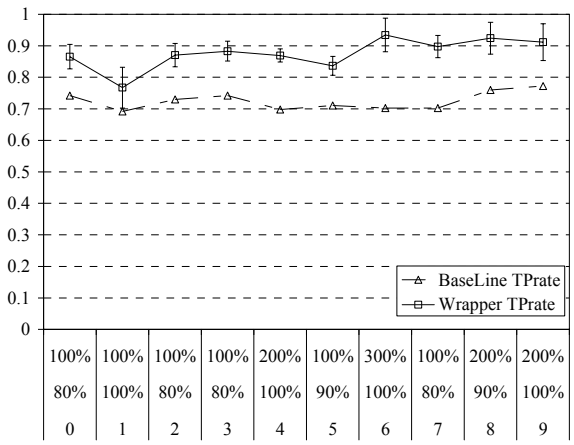

Figure 4-5. True Positive Rate for Phoneme Data Using RIPPER – SMOTE with Under-sampling

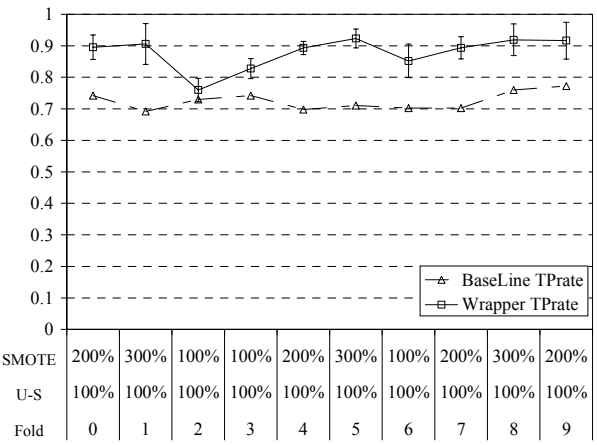

Figure 4-6. True Positive Rate for Phoneme Data Using RIPPER – SMOTE Only

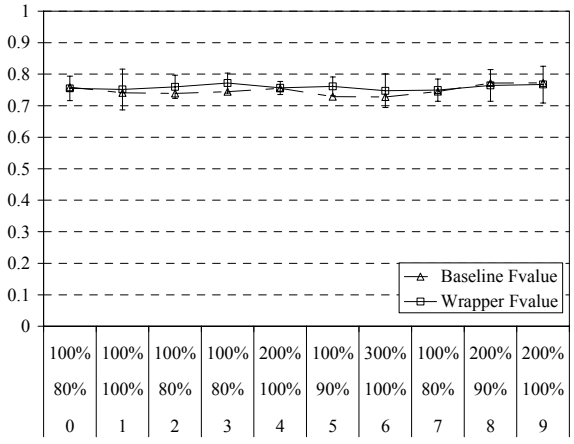

Figure 4-7. F-value for Phoneme Data Using RIPPER – SMOTE with Under-sampling

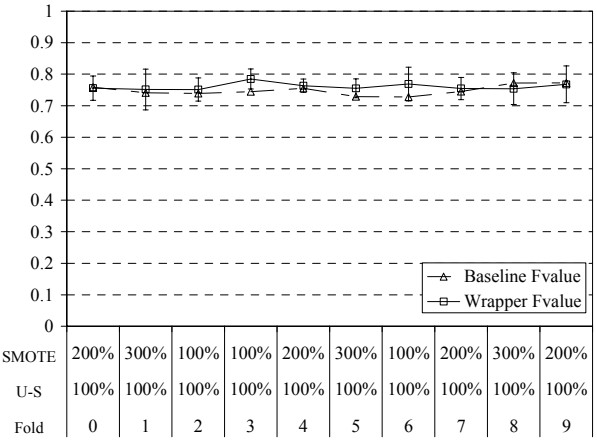

Figure 4-8. F-value for Phoneme Data Using RIPPER – SMOTE Only

From Figure 4-5 and Figure 4-6, it can be seen that the wrapper approach for RIPPER was able to significantly improve TP-rates (17.47% and 17.21% increase on average for 'SMOTE only' and 'under-sampling & SMOTE' respectively) on all 10 test sets in both scenarios with average TP-rate values very close to those obtained by C4.5. The wrapper F-values (1.62% and 1.33% increase on average for 'SMOTE only' and 'under-sampling & SMOTE' respectively, Figure 4-7 and Figure 4-8) for the minority class were statistically better than the baseline F-values. The wrapper F-values on the majority class were statistically worse than the baseline F-values as seen in Table 4-2.

4.5.2. Satimage Dataset

Originally, the Satimage Dataset [45] contained 6435 examples with 36 numeric attributes (4 spectral band values for each of nine pixels in a neighborhood) and six classes. But we chose the smallest class as the minority class and relabeled the rest of the classes as one majority class as was done in [7] [43]. This manipulation gave us a skewed two-class dataset, with 5809 samples in the majority class and 626 samples in the minority class. The results are shown in Table 4-3 below.

Table 4-3. Results for Satimage Dataset

	Algorithm	C4.5		RIPPER	
		SMOTE only	Under-sampling & SMOTE	SMOTE only	Under-sampling & SMOTE
	Average SMOTE percentage	630%	620%	440%	570%
	Average Under-sampling percentage	100%	93%	100%	87%
Average Minority class TP-rate	Baseline	0.538	0.538	0.505	0.505
	Wrapper	0.641	0.656	0.667	0.675
	% increase	15.99%	17.86%	24.33%	25.25%
	t-stat	-4.974	-5.389	-7.158	-6.791
	significance	√	√	√	√
Average Minority class F-value	Baseline	0.566	0.566	0.568	0.568
	Wrapper	0.569	0.566	0.596	0.586
	% increase	0.41%	0.01%	4.56%	3.07%
	t-stat	-0.145	-0.003	-1.926	-1.126
	significance	-	-	√	-
Average Majority class F-value	Baseline	0.956	0.956	0.96	0.96
	Wrapper	0.947	0.945	0.951	0.948
	% decrease	0.97%	1.15%	0.94%	1.23%
	t-stat	4.975	6.691	6.263	7.783
	significance	×	×	×	×

From Table 4-3, we can see that in all four experimental runs, the wrapper algorithm was able to significantly improve TP-rates for the minority class at the expense of a statistically significant reduction in F-values for the majority class. However, the amount of increase in the minority class TP rates was very large compared to the amount of reduction in the majority class F-values.

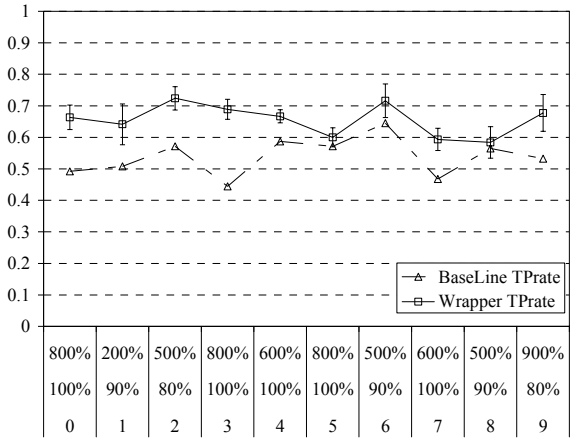

Figure 4-9. True Positive Rate for Satimage Data Using C4.5 – SMOTE with Under-sampling

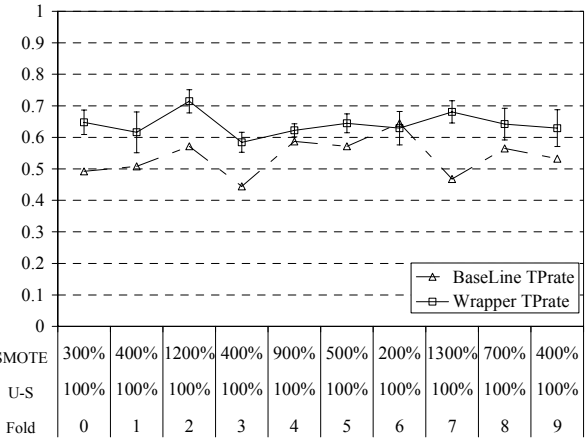

Figure 4-10. True Positive Rate for Satimage Data Using C4.5 – SMOTE Only

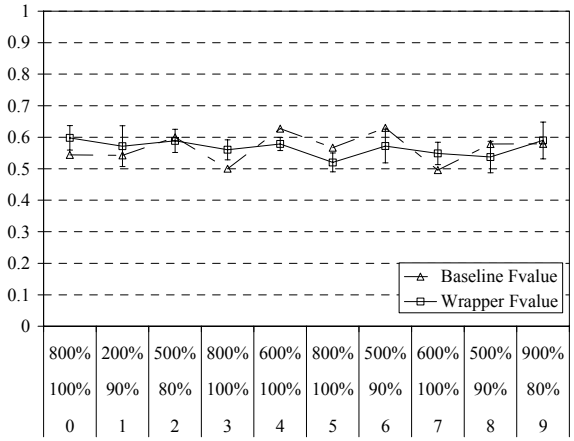

Figure 4-11. F-value for Satimage Data Using C4.5 – SMOTE with Under-sampling

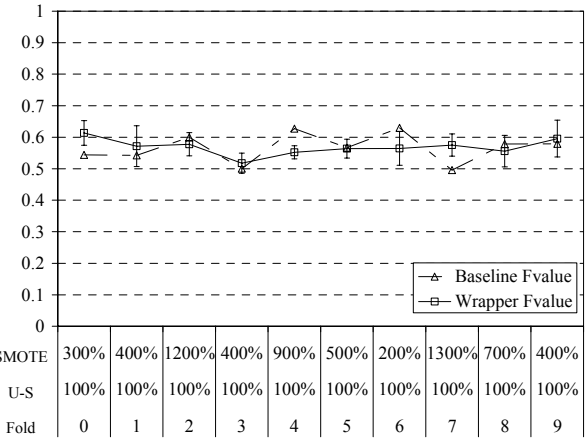

Figure 4-12. F-value for Satimage Data Using C4.5 – SMOTE Only

From Figure 4-9 and Figure 4-10, it can be seen that the wrapper approach for C4.5 was mostly able to significantly improve TP-rates for the minority class on 10 test sets, with the average wrapper TP-rate increase using 'under-sampling & SMOTE' (17.86%) better than average wrapper TP-rate using 'SMOTE only' (15.99%). The wrapper F-values for the minority class were statistically significantly different than the baseline F-values. As seen in Figure 4-11 and Figure 4-12, the wrapper F-values on the majority class were statistically significantly worse than the baseline F-values, but the amount of reduction was very small (0.97% and 1.15% on average for 'SMOTE only' and 'under-sampling & SMOTE' respectively).

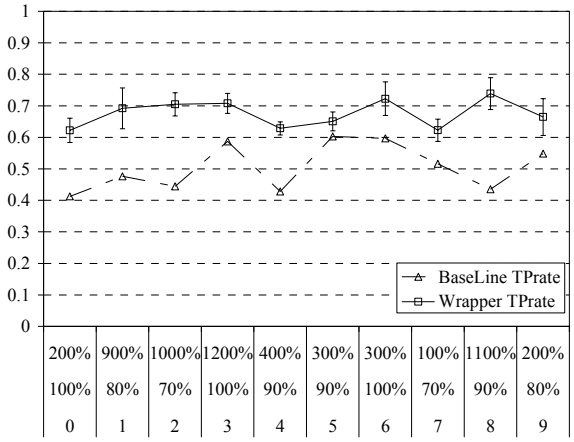

Figure 4-13. True Positive Rate for Satimage Data Using RIPPER – SMOTE with Under-sampling

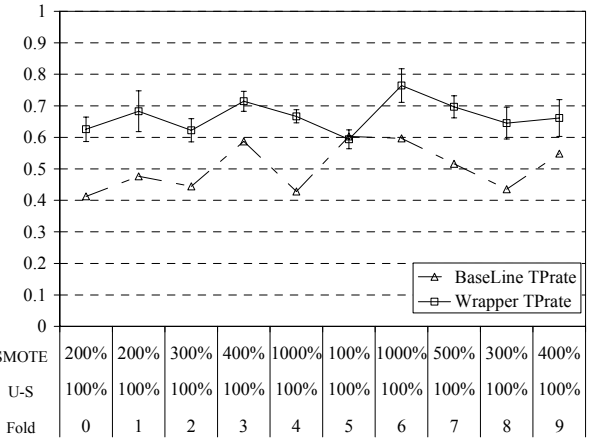

Figure 4-14. True Positive Rate for Satimage Data Using RIPPER – SMOTE Only

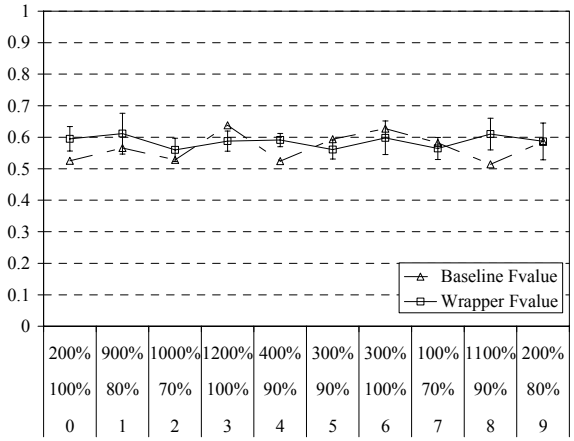

Figure 4-15. F-value for Satimage Data Using RIPPER – SMOTE with Under-sampling

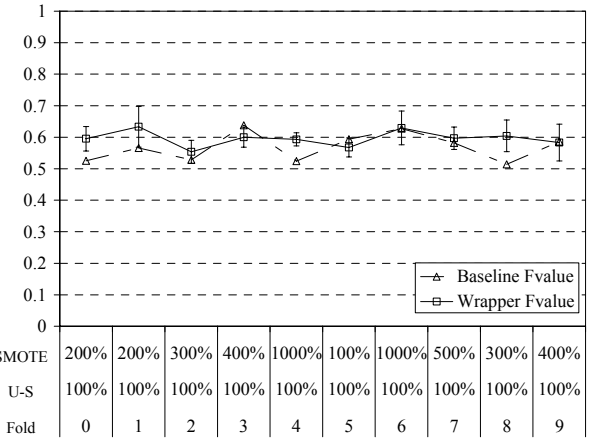

Figure 4-16. F-value for Satimage Data Using RIPPER – SMOTE Only

From Figure 4-13 and Figure 4-14, it can be seen that the wrapper approach for RIPPER was able to significantly improve TP-rates for the minority class on almost all 10 test sets, with the average wrapper TP-rate increases (24.33% and 25.25% on average for 'SMOTE only' and 'under-sampling & SMOTE' respectively) much higher than those obtained using C4.5 (15.99% and 17.86% for 'SMOTE only' and 'under-sampling & SMOTE' respectively). The wrapper F-values for the minority class were significantly improved for the 'SMOTE only' scenario but not for the 'under-sampling & SMOTE' scenario as seen in Table 4-3. The wrapper F-values for the majority class were statistically worse than the baseline F-values.

4.5.3. Mammography Dataset

The Mammography Dataset which was used in [8] consists of 11,183 total samples with six numeric features and two classes representing calcification (cancerous) and non-calcification (non-cancerous). The minority class which represents calcification contained only 260 examples in the dataset i.e. only 2.32% of the total examples. The results obtained are tabulated below in Table 4-4. The negative sign before the number in the '% increase' row indicates reduction in the value for that metric.

Table 4-4. Results for Mammography Dataset

	Algorithm	C4.5		RIPPER	
		SMOTE only	Under-sampling & SMOTE	SMOTE only	Under-sampling & SMOTE
	Average SMOTE percentage	210%	180%	300%	180%
	Average Under-sampling percentage	100%	87%	100%	94%
Average Minority class TP-rate	Baseline	0.546	0.546	0.577	0.577
	Wrapper	0.658	0.659	0.696	0.665
	% increase	16.96%	17.15%	17.13%	13.29%
	t-stat	-4.7	-3.913	-4.276	-3.322
	significance	√	√	√	√
Average Minority class F-value	Baseline	0.644	0.644	0.652	0.652
	Wrapper	0.647	0.634	0.643	0.639
	% increase	0.49%	-1.61%	-1.38%	-1.90%
	t-stat	-0.128	0.398	0.484	0.727
	significance	-	-	-	-
Average Majority class F-value	Baseline	0.993	0.993	0.993	0.993
	Wrapper	0.992	0.991	0.991	0.991
	% decrease	0.16%	0.21%	0.21%	0.18%
	t-stat	3.678	3.923	5.674	4.224
	significance	×	×	×	×

It can be seen from Table 4-4, that for all four experimental runs, the wrapper algorithm was able to statistically significantly improve TP-rates for the minority class at the expense of a statistically significant reduction in F-values for the majority class. Also the amount of increase in the minority class TP rates was very large as compared to the amount of reduction in the majority class F-values as shown in the '% increase' line.

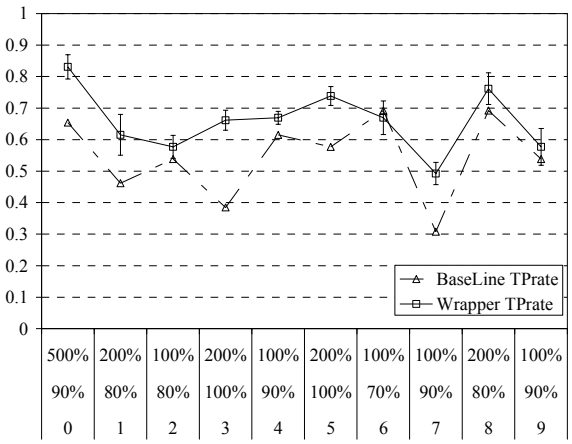

Figure 4-17. True Positive Rate for Mammography Data Using C4.5 – SMOTE with Under-sampling

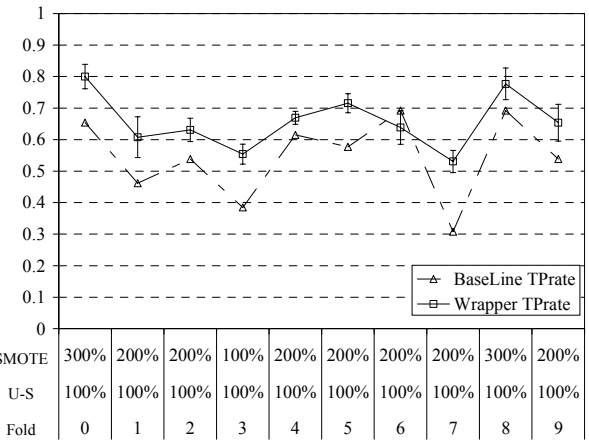

Figure 4-18. True Positive Rate for Mammography Data Using C4.5 – SMOTE Only

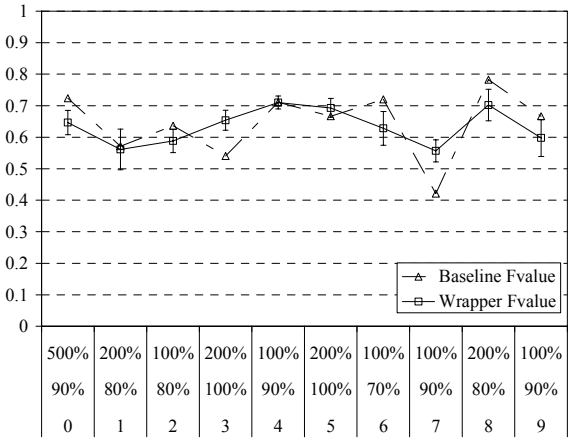

Figure 4-19. F-value for Mammography Data Using C4.5 – SMOTE with Under-sampling

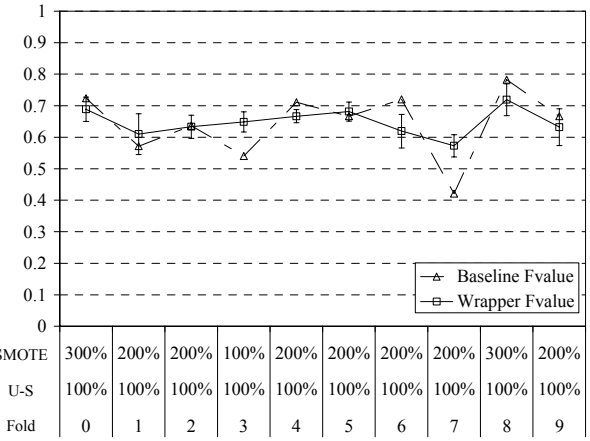

Figure 4-20. F-value for Mammography Data Using C4.5 – SMOTE Only

From Figure 4-17 and Figure 4-18 it can be seen that the wrapper approach for C4.5 was able to improve TP-rates for the minority class on 9 of 10 test sets. There was no statistically significant improvement in wrapper F-values for the minority class. Again the wrapper F-values on the majority class were statistically significantly worse than the baseline F-values, but the amount of reduction was extremely small (0.16% and 0.21% on average for 'SMOTE only' and 'under-sampling & SMOTE' respectively). These results show that statistical significance in the case of majority class F-values is sometimes not very important because the amount of reduction is very small and is worth the loss.

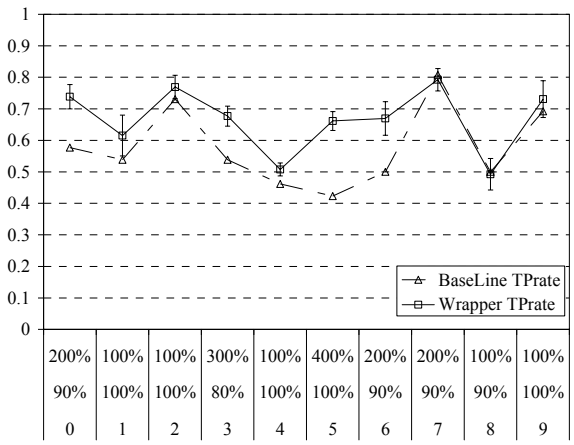

Figure 4-21. True Positive Rate for Mammography Data Using RIPPER – SMOTE with Under-sampling

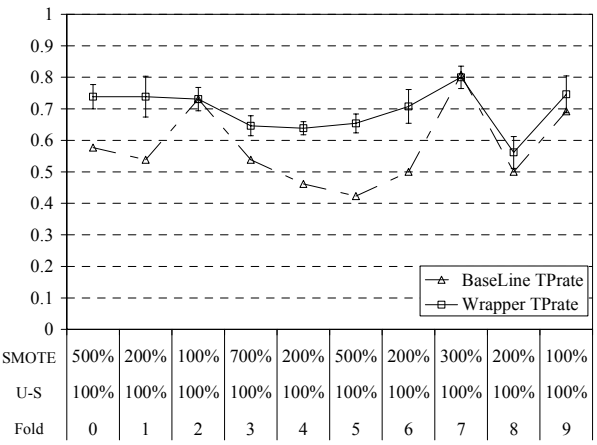

Figure 4-22. True Positive Rate for Mammography Data Using RIPPER – SMOTE Only

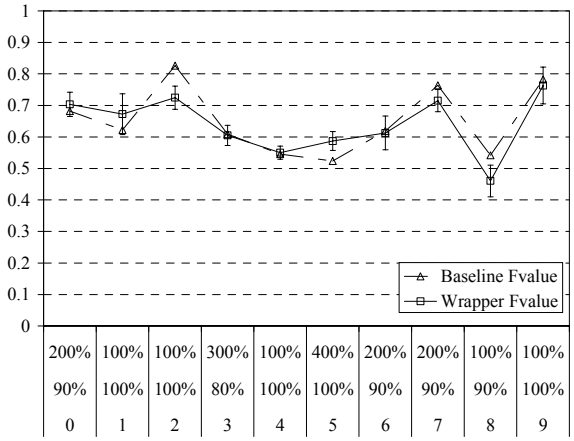

Figure 4-23. F-value for Mammography Data Using RIPPER – SMOTE with Under-sampling

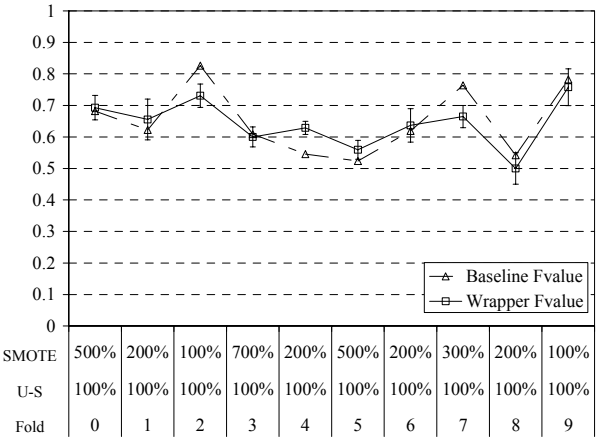

Figure 4-24. F-value for Mammography Data Using RIPPER – SMOTE Only

For RIPPER, as shown in Figure 4-21 and Figure 4-22, the wrapper F-values for the minority and majority class were very similar to those obtained using C4.5. The average TP-rate obtained by wrapper using the 'SMOTE only' (0.696) was higher than the average TP-rate got using the 'Under-sampling & SMOTE' (0.665) but there was no statistical difference between the two groups of results (t-stat = 1.63). One can also note that RIPPER performed better than C4.5 on the minority class (0.577 against 0.546 for TP-rate and 0.652 against 0.644 for F-value) when the imbalance in the data was large as in case of Mammography dataset (order of imbalance is 1:42). Whereas in the case of Phoneme dataset (order of imbalance is 1:2.4) where the imbalance was not large, C4.5 seemed to perform much better than RIPPER.

4.5.4. Forest Cover Dataset

Originally, the Forest Cover dataset [45] consisted of 581,012 examples with 54 numeric features related to cartographic variables and seven classes representing the type of the forest cover. For our study, the data samples from two classes were extracted while the rest were ignored as done in [7]. The two classes we considered are Ponderosa Pine with 35,754 samples and Cottonwood/Willow with 2,747 samples. The results obtained on this dataset are tabulated below in Table 4-5.

Table 4-5. Results for Forest Cover Dataset

	Algorithm	C4.5		RIPPER	
		SMOTE only	Under-sampling & SMOTE	SMOTE only	Under-sampling & SMOTE
	Average SMOTE percentage	600%	430%	560%	580%
	Average Under-sampling percentage	100%	99%	100%	93%
Average Minority class TP-rate	Baseline	0.873	0.873	0.834	0.834
	Wrapper	0.905	0.903	0.900	0.905
	% increase	3.59%	3.28%	7.34%	7.87%
	t-stat	-4.889	-4.223	-7.544	-7.532
	significance	√	√	√	√
Average Minority class F-value	Baseline	0.887	0.887	0.852	0.852
	Wrapper	0.884	0.885	0.868	0.866
	% increase	-0.35%	-0.24%	1.88%	1.69%
	t-stat	0.771	0.549	-3.963	-3.178
	significance	-	-	√	√
Average Majority class F-value	Baseline	0.992	0.992	0.989	0.989
	Wrapper	0.991	0.991	0.989	0.989
	% decrease	0.06%	0.05%	-0.06%	-0.04%
	t-stat	2.225	1.884	-2.19	-1.087
	significance	×	×	√	-

For the Forest cover dataset, the results for the minority class were as expected, with the wrapper TP-rate increasing with statistical significance. But the interesting thing about these results was that, the wrapper F-values obtained on the majority class using RIPPER in both scenarios had actually increased slightly instead of decreasing which was the general trend. For the 'SMOTE only' scenario using RIPPER, the wrapper F-values were better than baseline F-values with statistical significance. For C4.5, the drop in

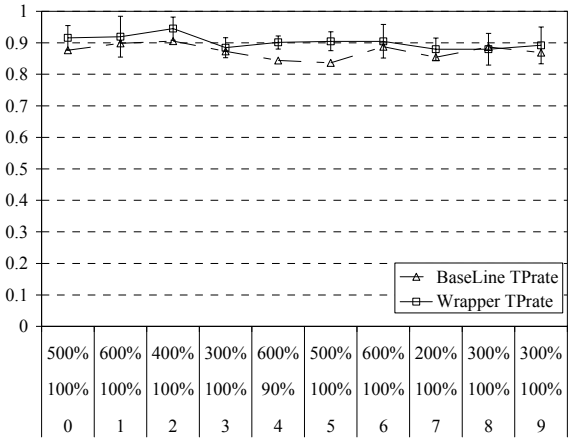

Figure 4-25. True Positive Rate for Forest cover Data Using C4.5 – SMOTE with Under-sampling

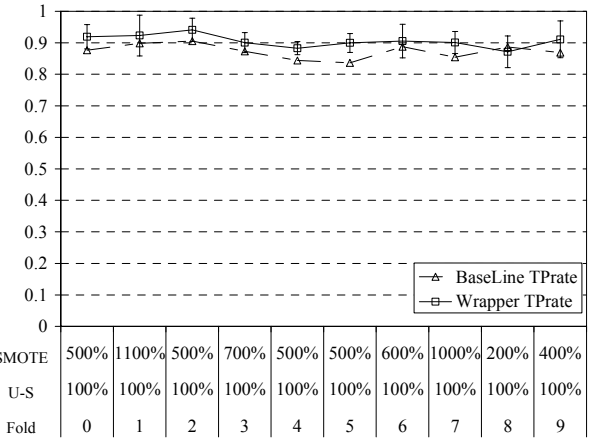

Figure 4-26. True Positive Rate for Forest cover Data Using C4.5 – SMOTE Only

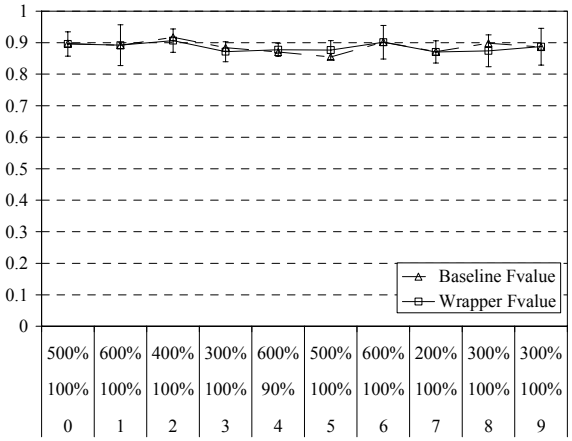

Figure 4-27. F-value for Forest cover Data Using C4.5 – SMOTE with Under-sampling

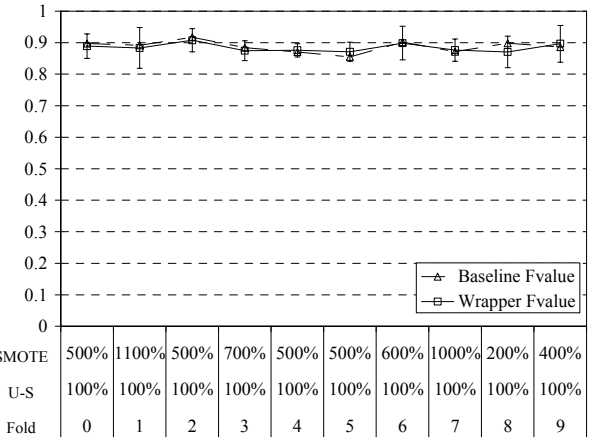

Figure 4-28. F-value for Forest cover Data Using C4.5 – SMOTE Only

the wrapper F-values over the majority class though statistically significant was extremely small. These are almost perfect results which one might always hope for, where the minority examples which were previously misclassified are correctly classified without increasing the number of majority class examples being classified as belonging to the minority class. The reason for these good results might be due to the similar distribution of the minority class examples in training and test data when cross-validation is performed. For example, the forest cover dataset which contains 2,747 total minority class examples, the training data will contain approximately 2,472 examples while test data will contain 275 examples. Whereas for other datasets like the Mammography and the oil dataset where the total number of minority

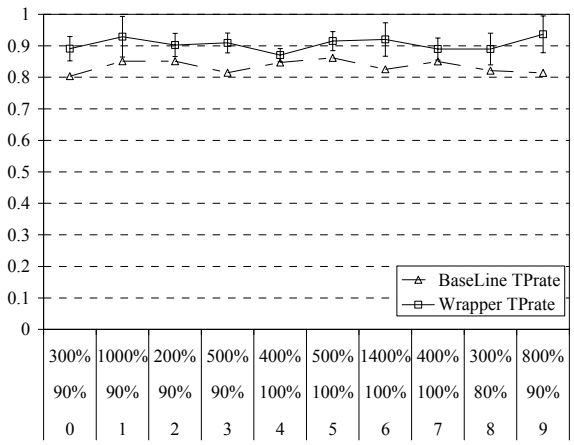

Figure 4-29. True Positive Rate for Forest cover Data Using RIPPER – SMOTE with Under-sampling

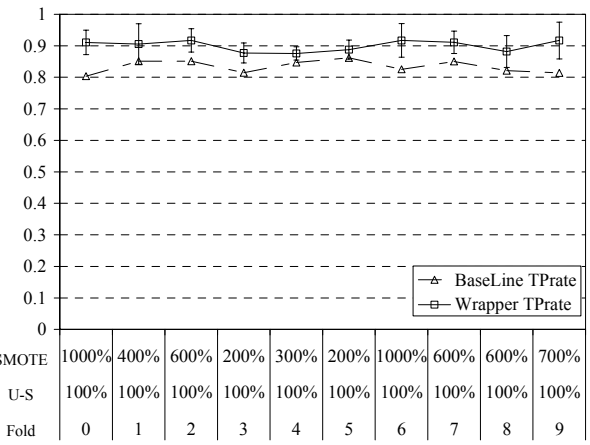

Figure 4-30. True Positive Rate for Forest cover Data Using RIPPER – SMOTE Only

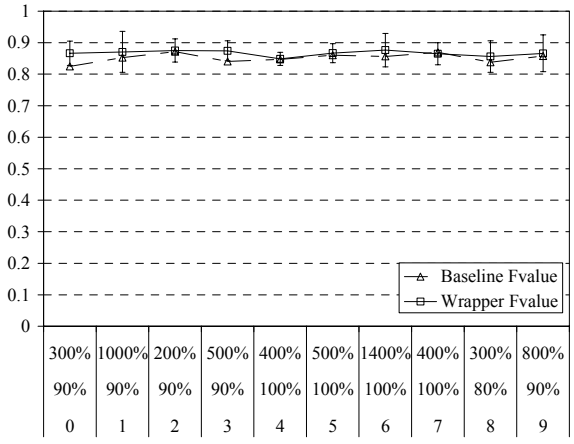

Figure 4-31. F-value for Forest cover Data Using RIPPER – SMOTE with Under-sampling

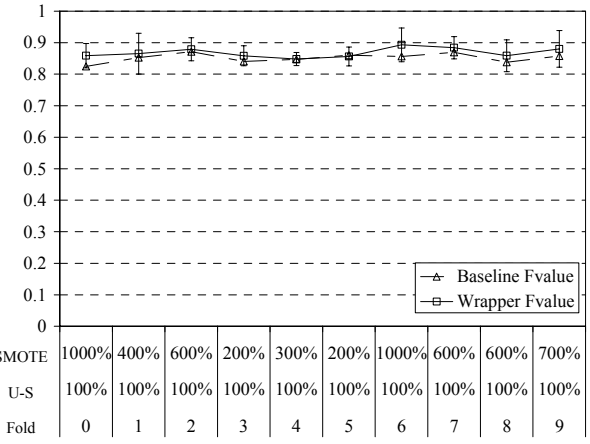

Figure 4-32. F-value for Forest cover Data Using RIPPER – SMOTE Only

class examples is very small (260 and 41 respectively), the training and the test datasets will contain a meager amount of minority class examples which might have a dissimilar distribution which may be lead to reduced accuracy on test data. So, it could be a small dataset phenomenon.

For RIPPER, from the Figure 4-29 and Figure 4-30 we can see that the wrapper approach was able to find under-sampling and SMOTE percentages which statistically significantly improve the TP-rates and F-values on the minority class and also the F-values on the majority class for the 'SMOTE only' scenario, though only by a small amount.

4.5.5. Pima Indian Diabetes Dataset

The Pima Indian Diabetes Dataset [45] was a result of survey examination done near Phoenix, Arizona, USA in accordance with World Health Organization criteria. It consists of 768 samples with 8 numeric variables and two classes. The aim of the dataset is to identify from the dependant variables whether a particular case is diabetic in nature. There are 268 cases which are diabetic which constitute the minority class while the remaining 500 cases belong to the non-diabetic majority class. The results are shown in Table 4-6 below.

Table 4-6. Results for Pima Indian Diabetes Dataset

	Algorithm	C4.5		RIPPER	
		SMOTE only	Under-sampling & SMOTE	SMOTE only	Under-sampling & SMOTE
	Average SMOTE percentage	350%	320%	300%	330%
	Average Under-sampling percentage	100%	89%	100%	89%
Average Minority class TP-rate	Baseline	0.608	0.608	0.59	0.59
	Wrapper	0.764	0.798	0.800	0.829
	% increase	20.32%	23.77%	26.27%	28.81%
	t-stat	-6.087	-5.756	-8.791	-8.734
	significance	√	√	√	√
Average Minority class F-value	Baseline	0.632	0.632	0.624	0.624
	Wrapper	0.635	0.643	0.643	0.650
	% increase	0.52%	1.72%	2.94%	3.94%
	t-stat	-0.145	-0.459	-0.888	-1.264
	significance	-	-	-	-
Average Majority class F-value	Baseline	0.814	0.814	0.816	0.816
	Wrapper	0.732	0.724	0.720	0.717
	% decrease	10.06%	10.98%	11.75%	12.09%
	t-stat	5.054	5.324	4.861	12.409
	significance	×	×	×	×

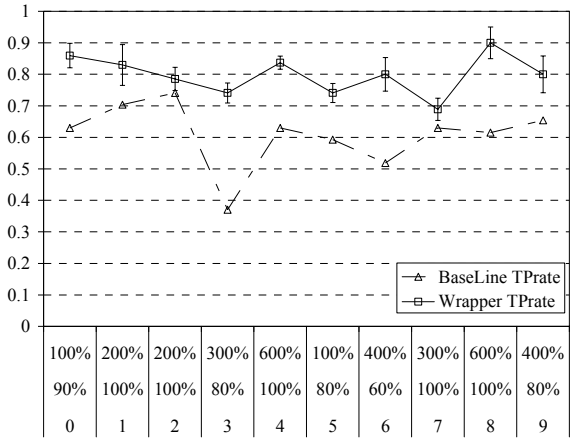

Figure 4-33. True Positive Rate for Pima Data Using C4.5 – SMOTE with Under-sampling

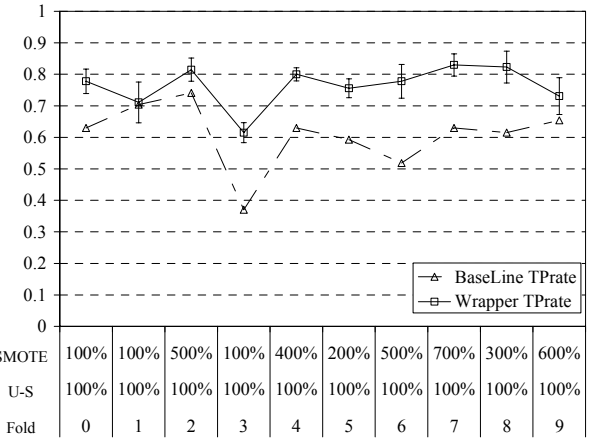

Figure 4-34. True Positive Rate for Pima Data Using C4.5 – SMOTE Only

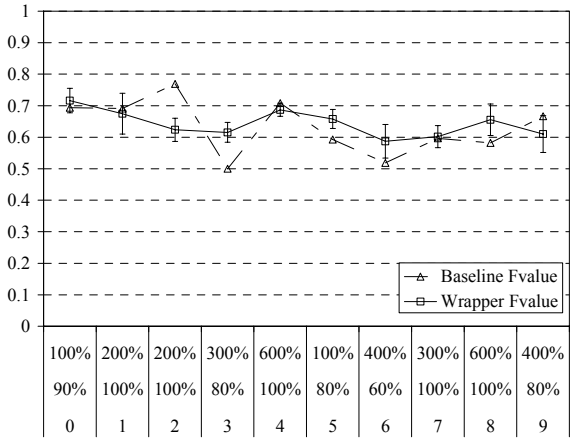

Figure 4-35. F-value for Pima Data Using C4.5 – SMOTE with Under-sampling

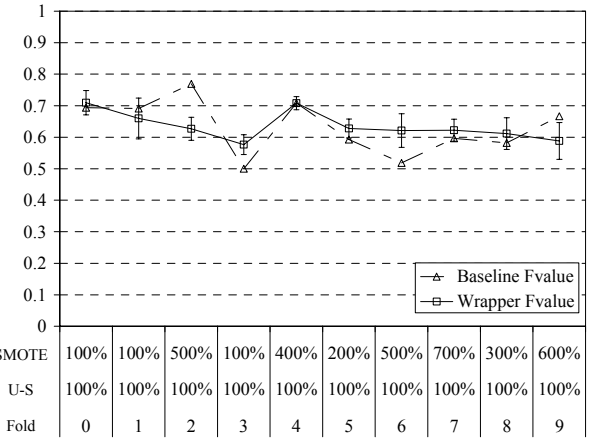

Figure 4-36. F-value for Pima Data Using C4.5 – SMOTE Only

For C.45, as can be seen from Table 4-6, Figure 4-35 and Figure 4-36, the wrapper TP-rates for both the scenarios were statistically significantly improved over the baseline TP-rates. There was a small increase in average F-value over the minority class but it isn't statistically significant. The wrapper F-values on the majority class were statistically significantly reduced. The average wrapper TP-rate and F-value on the minority class for the 'SMOTE only' scenario were lower than those obtained by the 'Under-sampling & SMOTE' scenario, but there was no statistically significant difference between the two results.

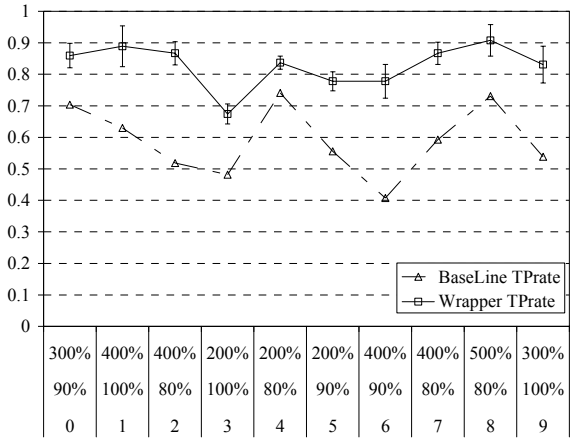

Figure 4-37. True Positive Rate for Pima Data Using RIPPER – SMOTE with Under-sampling

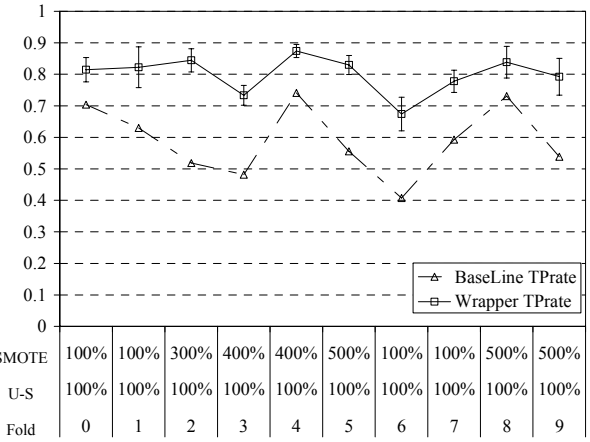

Figure 4-38. True Positive Rate for Pima Data Using RIPPER – SMOTE Only

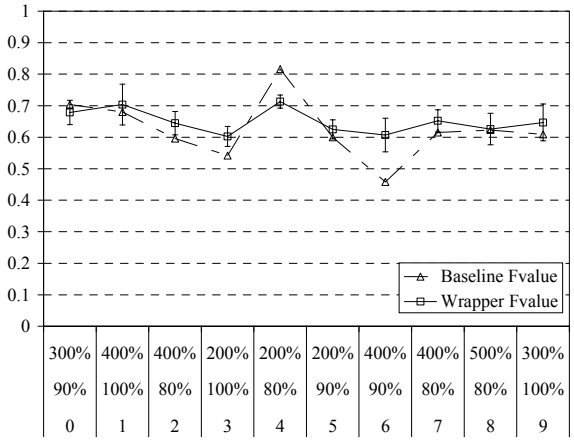

Figure 4-39. F-value for Pima Data Using RIPPER – SMOTE with Under-sampling

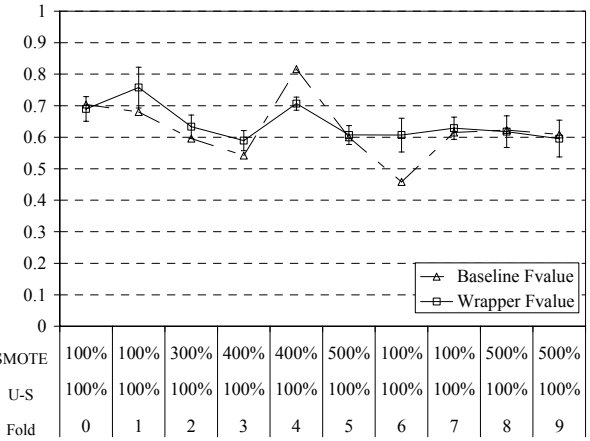

Figure 4-40. F-value for Pima Data Using RIPPER – SMOTE Only

From Table 4-6, Figure 4-37 and Figure 4-38, one can see that the wrapper TP-rates for both the scenarios using RIPPER were statistically significantly improved over the baseline TP-rates. There was no statistical significance to the improvement observed in wrapper F-value on minority class. The wrapper F-values on majority class were statistically significantly reduced when compared to baseline F-values. Also similar to results obtained using C4.5, the average wrapper TP-rate and F-value on the minority class for 'SMOTE only' scenario were lower than those obtained by 'Under-sampling & SMOTE' scenario, but again with no statistically significant difference between the two results.

4.5.6. Oil Dataset

The Oil dataset was provided by Robert Holte and was previously used in their paper [9]. This dataset has 41 oil slick samples and 896 non-oil slick samples. The results are shown in Table 4-7 below.

Table 4-7. Results for Oil Dataset

	Algorithm	C4.5		RIPPER	
		SMOTE only	Under-sampling & SMOTE	SMOTE only	Under-sampling & SMOTE
	Average SMOTE percentage	480%	370%	290%	340%
	Average Under-sampling percentage	100%	89%	100%	93%
Average Minority class TP-rate	Baseline	0.37	0.37	0.385	0.385
	Wrapper	0.487	0.520	0.555	0.529
	% increase	24.02%	28.85%	30.63%	27.22%
	t-stat	-2.885	-2.739	-2.457	-1.787
	significance	√	√	√	-
Average Minority class F-value	Baseline	0.408	0.408	0.435	0.435
	Wrapper	0.463	0.428	0.487	0.441
	% increase	11.70%	4.58%	10.70%	1.46%
	t-stat	-1.577	-0.542	-0.742	-0.09
	significance	-	-	-	-
Average Majority class F-value	Baseline	0.976	0.976	0.979	0.979
	Wrapper	0.974	0.969	0.973	0.969
	% decrease	0.50%	0.70%	0.63%	1.05%
	t-stat	0.816	2.897	2.28	3.156
	significance	-	×	×	×

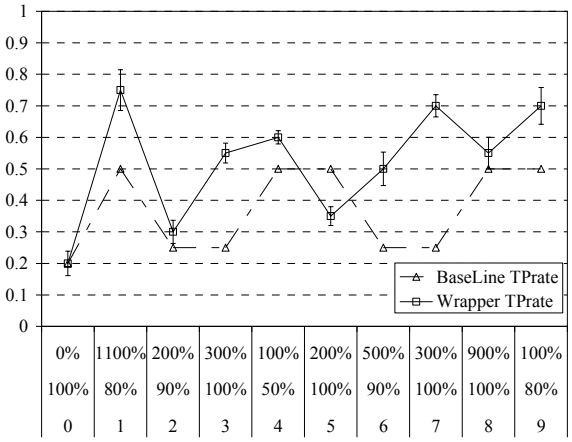

Figure 4-41. True Positive Rate for Oil Data Using C4.5 – SMOTE with Under-sampling

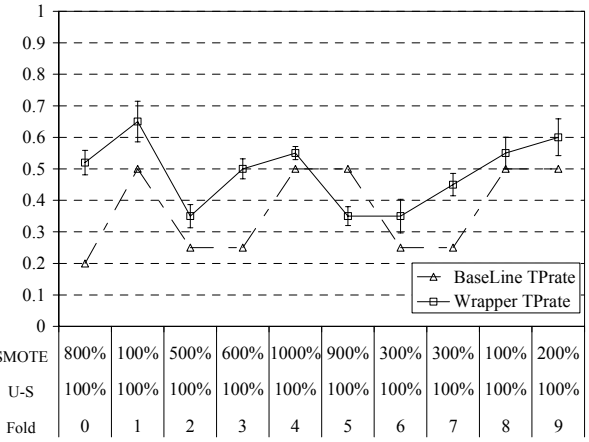

Figure 4-42. True Positive Rate for Oil Data Using C4.5 – SMOTE Only

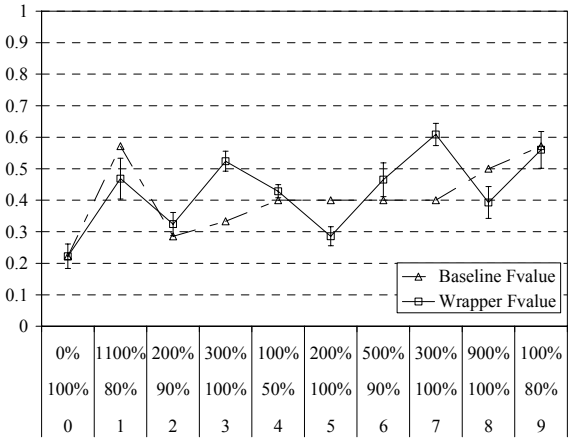

Figure 4-43. F-value for Oil Data Using C4.5 – SMOTE with Under-sampling

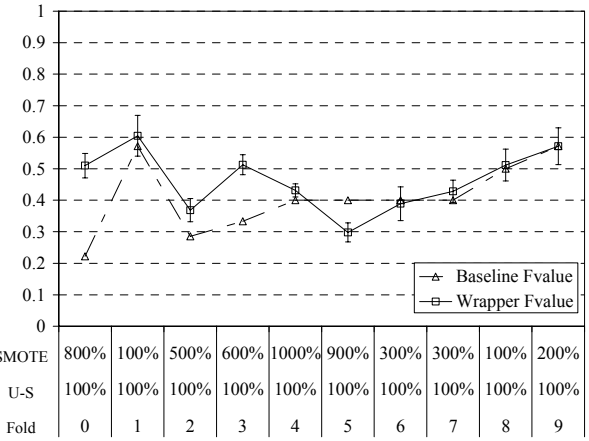

Figure 4-44. F-value for Oil Data Using C4.5 – SMOTE Only

From the Figure 4-41 to Figure 4-44, one can see there was lot of variation in the baseline and wrapper TP-rates and F-values across the 10 test folds. This might be due to poor representation of the minority class in the training data during 10 fold cross-validation where the training data containing approximately 37 examples might not be able to properly represent the test data containing approximately 4 examples. Still the wrapper TP-rates on the minority class were statistically better than baseline TP-rates. There was no statistical difference between wrapper and baseline F-values on the minority class. One can note that the average wrapper F-values on the minority and majority classes for the 'SMOTE only' scenario were better than those obtained by the 'Under-sampling & SMOTE' scenario.

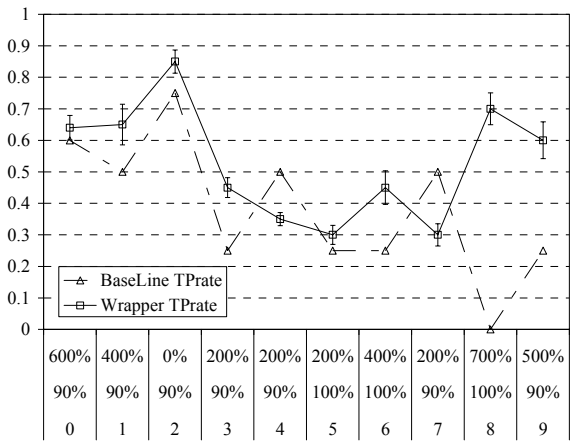

Figure 4-45. True Positive Rate for Oil Data Using RIPPER – SMOTE with Under-sampling

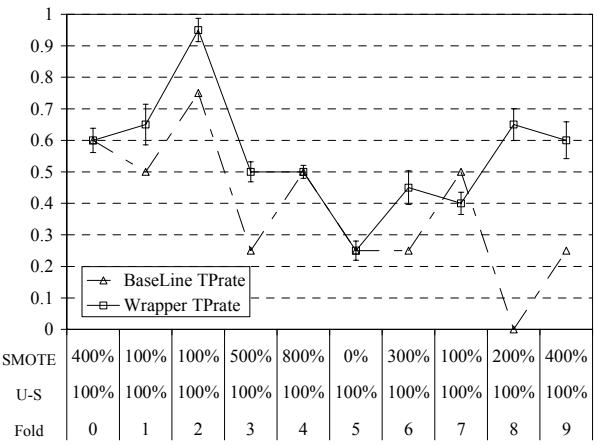

Figure 4-46. True Positive Rate for Oil Data Using RIPPER – SMOTE Only

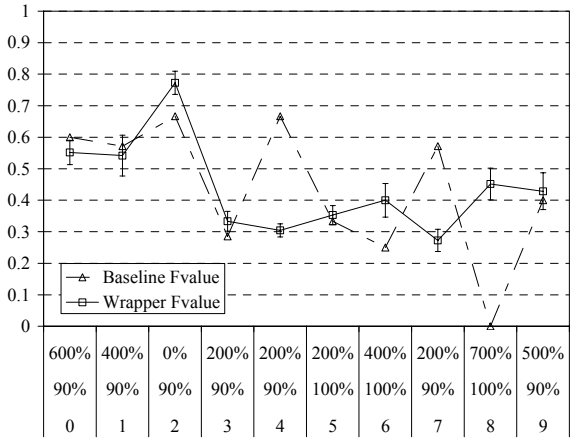

Figure 4-47. F-value for Oil Data Using RIPPER – SMOTE with Under-sampling

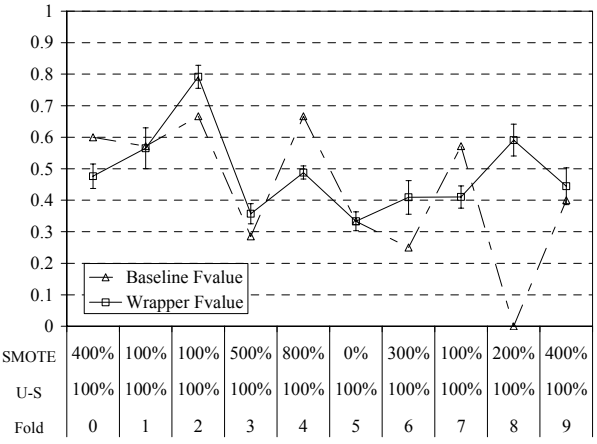

Figure 4-48. F-value for Oil Data Using RIPPER – SMOTE Only

The variation in results across the folds was even higher for RIPPER when compared with C.45. From Figure 4-45 to Figure 4-48, one can see that in fold 8, the baseline TP-rate and F-value on the minority class for the test data was zero. This might have been the result of RIPPER internally splitting the training data into growing and pruning sets, thus further reducing the number of minority class examples used to actually build the rules.

Due to the high variation in TP-rate across the 10 folds, in the 'Under-sampling & SMOTE' scenario where the average TP-rate on the minority class increases from 0.385 to 0.529, no statistical significance was detected in the difference between the wrapper and baseline TP-rates by the paired t-test.

4.5.7. KDD-cup 99: Network Intrusion Detection Dataset

The Network Intrusion Detection dataset was used in the KDD cup 99 competition [55] which was a modified version of the DARPA 1998 Intrusion Detection Evaluation Data [56] originally created by the MIT Lincoln Lab [57]. The task for the competition was to build a network intrusion detector using the data and identify the 'bad' connections or the attacks. The Dataset has 41 total features with 7 of its features nominal and the others continuous valued. The original training data contained 4,898,430 network connections while the test data contained 311,029 network connections. The original class labels are collapsed into five classes with the 'normal' class consisting of normal network connections while the other classes consisting of network attacks. Table 4-8 gives the mapping of the attacks to the class categories and the number of (total and distinct) examples in the training and test datasets. The four classes covering the four categories of attacks are –

- DOS (Denial of Service): ex. Ping-of-death, teardrop, smurf, etc.
- R2L (Remote to Local): unauthorized access from a remote machine, ex. Guessing password, etc.
- U2R (User to Remote): unauthorized access to local super-user privileges by local unprivileged user, ex. Buffer overflow attacks, etc.
- Probe: surveillance and probing, ex. Port-scan, ping-sweep, etc.

The test dataset has many novel attacks for 'u2r' (83% new attacks) and 'r2l' (63% new attacks) categories. Due to this totally different distribution of original training and test sets, we use two version of this dataset: one is a modified version of the dataset previously used by Chawla et al. [8] and the second is the original version used in the KDD cup competition. The modified dataset was created by merging the original training and testing sets and sampling 69,980 total examples. Under-sampling was performed only on the majority class (normal, dos and probe) examples with all the minority class (u2r and r2l) examples retained. Then this sampled dataset was randomly divided into two equal sets to create the new modified training and modified testing sets so that they had similar distributions. Since the datasets are big and separate testing sets are available, the evaluation of the wrapper Under-sample SMOTE algorithm is done using a testing set and not by using the 10-fold cross validation. Since we do have 10 paired results, the paired t-test is not performed for this dataset.

Table 4-8. Class Categories and Number of Examples in Original Training and Test Sets

		Training Data		Testing Data	
Class	Class Category	No. Examples	No. Distinct Examples	No. Examples	No. Distinct Examples
smurf	dos	2,807,886	3,007	164,091	936
neptune	dos	1,072,017	242,149	58,001	20,332
normal	normal	972,780	812,813	60,593	47,913
satan	probe	15,892	5,019	1,633	860
ipsweep	probe	12,481	3,723	306	155
portsweep	probe	10,413	3,564	354	174
nmap	probe	2,316	1,554	84	80
back	dos	2,203	968	1,098	386
warezclient	r2l	1,020	893		
teardrop	dos	979	918	12	12
pod	dos	264	206	87	45
guess_passwd	r2l	53	53	4,367	1,302
buffer_overflow	u2r	30	30	22	22
land	dos	21	19	9	9
warezmaster	r2l	20	20	1,602	1,002
imap	r2l	12	12	1	1
rootkit	u2r	10	10	13	13
loadmodule	u2r	9	9	2	2
ftp_write	r2l	8	8	3	3
multihop	r2l	7	7	18	18
phf	r2l	4	4	2	2
perl	u2r	3	3	2	2
spy	r2l	2	2		
apache2	dos			794	794
mailbomb	dos			5,000	308
processtable	dos			759	744
udpstorm	dos			2	2
mscan	probe			1,053	1,049
saint	probe			736	364
named	r2l			17	17
sendmail	r2l			17	15
snmpgetattack	r2l			7,741	179
snmpguess	r2l			2,406	359
worm	r2l			2	2
xlock	r2l			9	9
xsnoop	r2l			4	4
httptunnel	u2r			158	145
ps	u2r			16	16
sqlattack	u2r			2	2
xterm	u2r			13	13
total		4,898,430	1,074,991	311,029	77,291

Table 4-9. Modified KDD-cup 99 Intrusion Detection Test Dataset Summary

Class	No. of Examples	Percentage of Examples
u2r	136	0.4%
r2l	1982	5.7%
dos	13027	37.2%
probe	2445	7.0%
normal	17400	49.7%

Table 4-10. Results for Modified KDD-cup 99 Intrusion Detection Dataset Using C4.5

SMOTE only						Under-sampling & SMOTE				
Class/ SMOTE or Under-sample %	Metric	baseline	wrapper	% increase		Class/ SMOTE or Under-sample %	Metric	baseline	wrapper	% increase
u2r	TP-rate	0.794	0.837	5.10%		u2r	TP-rate	0.794	0.832	4.59%
200%	F-value	0.818	0.847	3.44%		0%	F-value	0.818	0.829	1.27%
r2l	TP-rate	0.968	0.973	0.53%		r2l	TP-rate	0.968	0.973	0.45%
800%	F-value	0.964	0.958	-0.62%		500%	F-value	0.964	0.958	-0.60%
dos	TP-rate	0.998	0.997	-0.12%		dos	TP-rate	0.998	0.997	-0.13%
100%	F-value	0.998	0.997	-0.07%		100%	F-value	0.998	0.997	-0.08%
probe	TP-rate	0.984	0.994	0.99%		probe	TP-rate	0.983	0.994	1.08%
100%	F-value	0.983	0.984	0.12%		100%	F-value	0.983	0.984	0.05%
normal	TP-rate	0.993	0.992	-0.14%		normal	TP-rate	0.993	0.992	-0.16%
100%	F-value	0.994	0.993	-0.08%		90%	F-value	0.994	0.993	-0.10%

Table 4-11. Results for Modified KDD-cup 99 Intrusion Detection Dataset Using RIPPER

SMOTE only						Under-sampling & SMOTE				
Class/ SMOTE or Under-sample %	Metric	baseline	wrapper	% increase		Class/ SMOTE or Under-sample %	Metric	baseline	wrapper	% increase
u2r	TP-rate	0.882	0.882	0.00%		u2r	TP-rate	0.882	0.862	-2.39%
400%	F-value	0.873	0.830	-5.09%		200%	F-value	0.873	0.855	-2.09%
r2l	TP-rate	0.967	0.968	0.18%		r2l	TP-rate	0.967	0.97	0.38%
100%	F-value	0.963	0.965	0.15%		100%	F-value	0.963	0.961	-0.21%
dos	TP-rate	0.998	0.998	-0.02%		dos	TP-rate	0.998	0.998	0.00%
100%	F-value	0.998	0.998	0.00%		90%	F-value	0.998	0.998	-0.03%
probe	TP-rate	0.993	0.986	-0.68%		probe	TP-rate	0.985	0.994	0.89%
100%	F-value	0.984	0.981	-0.30%		100%	F-value	0.984	0.983	-0.12%
normal	TP-rate	0.992	0.992	0.01%		normal	TP-rate	0.992	0.992	-0.04%
100%	F-value	0.994	0.994	-0.01%		100%	F-value	0.994	0.993	-0.05%

As indicated earlier in the wrapper Under-sample SMOTE algorithm, the greedy search for the under-sampling begins with no under-sampling for all the majority classes, then only the 'normal' class in under-sampled at 90%. If results after under-sampling do not satisfy the stopping conditions, then the 'dos' along with 'normal' class are under-sampled at 90% and the search continues. If for any of the majority classes the stopping condition is met (i.e. either the average F-value on the minority classes is reduced or the average F-value on the majority classes is reduced by 5%), the under-sampling for that majority class is reset to the previous under-sampling value and the search for under-sampling for any other majority classes which have not met the stopping conditions continues. A similar search for SMOTE percentages for the minority classes which also includes a look ahead is performed. Table 4-9 gives a summary of the modified KDD cup 99 Intrusion Detection test dataset. Table 4-10 and Table 4-11 show the results obtained on all minority and majority classes using the C4.5 and RIPPER learning algorithms. In the tables, below the name of each class, SMOTE percentage or under-sampling percentage for the minority class (u2r and r2l) or majority class (normal, dos and probe) selected by the wrapper algorithm are shown.

From Table 4-10, one can see when using the C4.5 learning algorithm, the wrapper TP-rates and F-values on the 'u2r' minority class, which was just 0.4% of the total test data, improved over the baseline results for both the 'SMOTE only' and 'Under-sampling & SMOTE' scenario. For the 'r2l' minority class, which was quite large compared to 'u2r' class (5.7% of total test data), the TP-rate increased while the F-value decreased in both the scenarios. It's interesting to see that for the 'probe' majority class which accounted for 37.2% of the test data, the TP-rate went up by around 1% in both scenarios. The TP-rates and the F-values for the 'dos' and 'normal' class went down by small amounts. Another interesting thing about the results is that, though 'u2r' was a very small class, its SMOTE percentage was much less than the 'r2l' class's SMOTE percentage.

For the RIPPER learning algorithm, the wrapper TP-rates and F-values for the 'u2r' class reduced instead of increasing. One can see in Table 4-11, that the baseline TP-rates and F-values for the 'u2r' minority class were very high when compared to C4.5 baseline as well as wrapper results. RIPPER which starts by building the rules for the smallest class first seemed to over-fit the training data when the 'u2r' class was SMOTEd. Also in the 'SMOTE only' scenario where the TP-rate (recall) for 'u2r' remained constant, the F-value went down by 5.09% due to reduction in precision over the test data which shows that

more examples from other classes were classified as belonging to 'u2r' class. The TP-rates and F-values obtained on other classes were similar to those obtained using C4.5.

For the evaluation on original version of the dataset, we kept the testing data untouched. Building a model on the original training data is very time consuming as it contains nearly 5 million records. So as done by many other researchers [58], we kept all the distinct examples for the category classes of 'u2r', 'r2l', 'probe', 'dos' and 'normal'. For the 'neptune' (dos) and 'normal' classes, there were still a huge number of distinct class examples (242,149 and 812,813 respectively). Many previous research studies [58] had done random under-sampling of the distinct 'normal' and 'dos' (neptune) class examples to further reduce the size of the training data. This strategy must have removed some of the important examples which might have occurred more than once in the original training data making the built classifier weaker. What we did was associate a count with each distinct record which counts the number of its occurrences in the original training data. Then by using the simple logic that 'a normal connection occurring only once in the dataset seems to be abnormal' we further under-sampled the 'normal' class to 34,397 examples which effectively represent 194,364 normal examples. Similarly, the 'neptune' class was further under-sampled by removing 155,549 examples which occurred only once with the assumption that some of them may be mislabeled, thus effectively representing 916,468 of 1,072,017 total 'neptune' examples. The details of the two versions of training datasets is shown are Table 4-12.

Table 4-12. Details of Two Versions of Original Training Data

Datasets	Original Training Data		Training Data 1 - under-sampled normal class		Training Data 2- under-sampled normal and Neptune	
	No. Examples	Ratio to Normal	No. Examples	Ratio to Normal	No. Examples	Ratio to Normal
normal	972,780	100.00%	34,397	100.00%	34,397	100.00%
probe	41,102	4.23%	13,860	40.29%	13,860	40.29%
dos	3,883,370	399.20%	247,267	718.86%	91,718	266.65%
r2l	1,126	0.12%	999	2.90%	999	2.90%
u2l	52	0.01%	52	0.15%	52	0.15%
Total	4,898,430		296,575		141,026	
Effective	4,898,430		4,120,014		3,964,465	

The KDD-cup competition used a cost matrix for evaluating the results on the testing set obtained by its participants, which is given in the Table 4-13.

Table 4-13. Cost Matrix Used for Scoring Entries in KDD Cup 99 Competition

Actual\predicted	dos	u2r	r2l	probe	normal
dos	0	2	2	1	2
u2r	2	0	2	2	3
r2l	2	2	0	2	4
probe	2	2	2	0	1
normal	2	2	2	1	0

Since a form of under-sampling was already performed, we ran our wrapper Under-sample SMOTE algorithm using RIPPER and C4.5 to search for SMOTE percentage only for the minority classes 'u2r' and 'r2l' on the two variants of the datasets – Training Data 1 for which only the 'normal' class was under-sampled and Training Data 2 for which the 'normal' and 'neptune' class were under-sampled. Table 4-14 shows the selected SMOTE percentages for the minority classes.

Table 4-14. SMOTE Percentages Selected by Under-sample SMOTE Algorithm

minority classes	Training Data 1 - under-sampled normal class		Training Data 2- under-sampled normal and Neptune	
	C4.5	RIPPER	C4.5	RIPPER
u2r	200%	100%	200%	100%
r2l	0%	0%	300%	0%

Table 4-15. Comparison of Results Obtained on Original KDD Cup 99 Test Data

	dos	u2r	r2l	probe	normal	Cost per test example
Winning Strategy [58]	97.10%	13.20%	8.40%	83.30%	99.50%	0.2331
Decision Tree[59]	96.57%	13.60%	0.45%	77.92%	99.43%	0.2371
Naïve Bayes [59]	96.65%	10.96%	8.66%	88.33%	97.68%	0.2485
Multi-classifier [60]	97.30%	29.80%	9.60%	88.70%	-	0.2285
Using C4.5 on Training Data 1 u2r (200) - r2l (0)	97.08%	14.47%	1.21%	93.52%	97.87%	0.2478
Using RIPPER on Training Data 1 u2r (100) - r2l (0)	97.45%	22.37%	6.96%	81.64%	96.18%	0.2444
Using C4.5 on Training Data 2 u2r (200) - r2l (300)	99.41%	14.47%	7.39%	93.61%	97.34%	0.2051
Using RIPPER on Training Data 2 u2r (100) - r2l (0)	97.33%	19.74%	13.73%	91.98%	95.62%	0.2049

For comparing the results, we use the original cost matrix to calculate the average cost per test example and the percentage correct examples for each class. Table 4-15 shows the comparison of the

results obtained by classifiers built using wrapper selected SMOTE percentages for 'u2r' and 'r2l' classes with the KDD cup 99 winner [58] and two other recent models [59] [60]. The dash symbol in the cell for Multi-classifier 'normal' class indicates a missing result. In [59] the author tried to show that a simple Naïve Bayes classifier was able to get very competitive results when compared to the winning KDD cup entry and showed an increase in accuracy over 'r2l' and 'probe' class. The average cost per test example was not provided by [59] for their two classifiers, so we calculated them from the confusion matrix they had provided. The values indicate that both the classifiers were actually weaker than the winning entry. In [60] a multi-classifier model was built using a Multi-Layered Perceptron for 'probe' class, K-means clustering for 'dos' and 'u2r' class and a Gaussian classifier for 'r2l' class. They were able to reduce the average cost per test example when compared to winning strategy and got the best accuracy on the 'u2r' class of 29.8%. Our two classifiers built on Training Data 1 i.e. with only the 'normal' class under-sampled and wrapper selected SMOTE for minority classes were worse than the winning strategy. The number of 'normal' examples were about $1/7^{th}$ of the number of 'dos' examples which might be the reason for low accuracy. The two classifiers (C4.5 and RIPPER) built on Training Data 2 which was created by under-sampling both the 'normal' and 'neptune' class where the ratio of 'dos' to 'normal' was more similar to original training data and SMOTEd at wrapper selected SMOTE percentages for minority classes were able to significant reduce the average cost per test example.

The same technique which was used in [58] was used to establish statistical significance for our results. The average cost per test example for our two classifiers (C4.5 and RIPPER) was 0.2051 and 0.2049 and the standard deviation was 0.8632 and 0.8175 respectively. The test dataset contains total 311,029 examples, but not all them were independent. An upper bound on the number of independent test examples is the number of distinct test examples, which is 77,291. So the standard error of our two models is at least 0. 8632/sqrt(77291) = 0.0031 and 0. 8175/sqrt(77291) = 0. 0029. Using two standard errors as threshold for statistical significance [58], both classifiers built on Training Data 2 with SMOTE were statistically significantly better than all other models. Also our models were able to get the best accuracies on 'dos', 'probe' and 'r2l' class so far. Thus these results on the original KDD-cup data show that clever under-sampling and SMOTE can be very useful in boosting accuracies over minority classes.

4.6. Summary

The results obtained on the various datasets are summarized in Table 4-16, which shows when statistical significance between the baseline and wrapper TP-rates for the minority class was observed. The symbol '√' indicates that wrapper TP-rates are statistically significantly better than baseline TP-rates whereas '-' indicates that there is no statistically significant difference between the two result groups. The table shows that the wrapper approach was able to statistically significantly improve the TP-rate of the minority class in almost all cases except for the oil dataset using RIPPER for the 'Under-sampling & SMOTE' scenario where the average wrapper TP-rate was better than the average baseline TP-rate but did not had any statistical significance. The F-values on minority classes were improved by a small amount in many cases but statistical significance was only occasionally observed.

Table 4-16. Significance of Improvement in TP-rate Over Minority Class

Dataset	% of Minority Class Data	C4.5		RIPPER	
		SMOTE only	Under-sampling & SMOTE	SMOTE only	Under-sampling & SMOTE
Phoneme	29.35%	√	√	√	√
Satimage	9.73%	√	√	√	√
Mammography	2.32%	√	√	√	√
Forest cover	7.13%	√	√	√	√
Pima Indian	34.90%	√	√	√	√
Oil	4.38%	√	√	√	-

Table 4-17. Statistical Significance Between 'Smote Only' & 'Under-sampling & SMOTE'

	C4.5 - 'SMOTE only' Vs. 'Under-sampling & SMOTE'			RIPPER - 'SMOTE only' Vs. 'Under-sampling & SMOTE'		
	Average Minority class TP-rate	Average Minority class F-value	Average Majority class F-value	Average Minority class TP-rate	Average Minority class F-value	Average Majority class F-value
Phoneme	-0.009(-)	0.007(-)	0.007(√)	0.003(-)	0.003(-)	0.001(-)
Satimage	-0.015(-)	0.003(-)	0.002(-)	-0.008(-)	0.010(√)	0.003(√)
Mammography	-0.001(-)	0.013(-)	0.001(-)	0.031(-)	0.004(-)	0.000(-)
Forest cover	0.002(-)	-0.001(-)	0.000(-)	-0.005(-)	0.002(-)	0.000(-)
Pima Indian	-0.034(-)	-0.008(-)	0.008(-)	-0.029(-)	-0.007(-)	0.003(-)
Oil	-0.033(-)	0.035(-)	0.005(√)	0.026(-)	0.046(-)	0.004(-)

Table 4-17 shows the difference between the absolute values of the wrapper performance metrics obtained by the 'SMOTE only' and 'Under-sampling & SMOTE' scenarios for individual inductive learning algorithms. A positive difference implies that the 'SMOTE only' approach was better than the 'Under-sampling & SMOTE' approach. The round brackets include a symbol which tells whether the difference had any statistical significance ('√' means that wrapper results are statistically significantly better than baseline results whereas '-' means that there is no statistically significant difference between the two result groups). One can see that in most of the cases there was no statistical difference between the wrapper results from the 'Smote Only' and 'Under-sampling & SMOTE' scenarios. When using the C4.5 learning algorithm, the wrapper F-value over the majority class was statistically better for the Phoneme dataset (though the difference was extremely small: 0.007) and the Oil dataset (the difference was extremely small: 0.005) when under-sampling of the majority class was not performed. Also in case of RIPPER, the wrapper F-value on the minority and majority class for the 'SMOTE only' scenario were statistically better than those obtained by 'Under-sampling & SMOTE' scenario on the Satimage dataset. The 'Under-sampling & SMOTE' approach seems to get a better TP-rate on the minority class than 'SMOTE only' while 'SMOTE only' seems to get better F-values on both the minority and majority classes. From this we can say that there was no clear benefit to under-sampling the majority class other than it helped reduce the learning time by reducing the dataset size.

Table 4-18 shows which of the two machine learning algorithms – C4.5 or RIPPER – got the best baseline and wrapper results for the two 'SMOTE only' and 'Under-sampling & SMOTE' approaches. The appropriate cell contains the name of the learning algorithm which came out to have the highest performance metric for the scenario with significance level included inside the round bracket. The symbol '√' indicates that the performance metric obtained by the selected learning algorithm was statistically significantly better than that obtained by the other learning algorithm.

From Table 4-18, it can observed that for the baseline results, RIPPER generally performs better than C4.5 on extremely skewed datasets like the Mammography, oil and KDDcup-99 datasets while C4.5 performs better than RIPPER for the reverse case. C4.5 builds a tree for the given dataset and effectively creates rules for all the classes in the dataset. While RIPPER which starts by building rules for a class which has the smallest number of examples leaves the largest class as default. So in case of two class

datasets, RIPPER only builds rules for the minority class. This seems to make RIPPER more biased towards the minority class when compared to C4.5 which helps in getting a better TP-rate on minority class when the order of imbalance is high. After under-sampling and SMOTEing or just SMOTEing, RIPPER generally gets a higher TP-rate for the minority class while C4.5, which is less biased towards the minority class, gets better F-values as in case of the Phoneme, Forest Cover and modified KDDcup-99 datasets.

Table 4-18. Winners: C4.5 Against RIPPER

Dataset	Performance metric	Baseline Results	Wrapper Results	
			SMOTE only	Under-sampling & SMOTE
Phoneme	Minority TP-rate	C4.5(√)	RIPPER(-)	RIPPER(-)
	Minority F-value	C4.5(√)	C4.5(√)	C4.5(√)
	Majority F-value	C4.5(-)	C4.5(√)	C4.5(√)
Satimage	Minority TP-rate	C4.5(-)	RIPPER(-)	RIPPER(-)
	Minority F-value	RIPPER(-)	RIPPER(√)	RIPPER(√)
	Majority F-value	RIPPER(√)	RIPPER(√)	RIPPER(√)
Mammo-graphy	Minority TP-rate	RIPPER(-)	RIPPER(-)	RIPPER(-)
	Minority F-value	RIPPER(-)	C4.5(-)	RIPPER(-)
	Majority F-value	C4.5(-)	C4.5(-)	RIPPER(-)
Forest cover	Minority TP-rate	C4.5(√)	C4.5(-)	RIPPER(-)
	Minority F-value	C4.5(√)	C4.5(√)	C4.5(√)
	Majority F-value	C4.5(√)	C4.5(√)	C4.5(√)
Pima Indian	Minority TP-rate	C4.5(-)	RIPPER(-)	RIPPER(-)
	Minority F-value	C4.5(-)	RIPPER(-)	RIPPER(-)
	Majority F-value	RIPPER(-)	C4.5(-)	C4.5(-)
Oil	Minority TP-rate	RIPPER(-)	RIPPER(-)	RIPPER(-)
	Minority F-value	RIPPER(-)	RIPPER(-)	RIPPER(-)
	Majority F-value	RIPPER(-)	C4.5(-)	RIPPER(-)
KDDcup-99 Intrusion Detection	Avg. Minority TP-rate	RIPPER	RIPPER	RIPPER
	Avg. Minority F-value	RIPPER	C4.5	RIPPER
	Avg. Majority F-value	RIPPER	C4.5	RIPPER

4.7. Result Comparison

In this subsection we compare the results obtained by our Wrapper Under-sample SMOTE Algorithm with those obtained in Chawla et al. [7] using the C4.5 decision tree classifier. Chawla et al. [7] used the Area Under the Curve (AUC) for ROC curves as a performance metric in the evaluation of the SMOTE technique. Note that they had hand picked the SMOTE percentage which gave them the best AUC measure over the 10 fold cross-validation test sets whereas in our case the test sets are never used in the search process for the under-sampling and SMOTE percentages. For generation of the ROC curves, a constant SMOTE percentage over all the 10 training sets was used and the under-sampling of the majority class was varied from 0% (no under-sampling) to 100% (zero majority class examples in training set) to obtain different points for the ROC curve. A particular under-sampling percentage was never sought in Chawla et al. [7] and was performed at irregular steps which were inconsistent across all the datasets, and so it was not possible to identify what under-sampling was used for a particular point on the ROC curve. Since we use the F-value as the performance metric, we have calculated the F-value for each point on the ROC curve from the TP rate, FP rate and the number of minority and majority class examples. In our case, since the under-sampling and SMOTE percentages are optimized for each fold, they differ across the 10 folds. So it was not reasonable to compare the average F-values over our 10 cross-validation folds with the F-values calculated from the ROC curves. To make the comparison more compatible we have re-ran the 10 folds using the 'mode' of the under-sampling and SMOTE percentages obtained by our wrapper algorithm and obtained an averaged F-value over 10 test sets. Table 4-19 and Table 4-20 show the comparison of our new results with the hand picked ones obtained by Chawla et al. [7].

From Table 4-19, one can see that the 'mode' of the SMOTE percentages selected by our wrapper Under-sample SMOTE algorithm match with the handed-picked SMOTE percentages for the Phoneme and the Forest cover datasets. Also the minority class average F-values obtained by the two methods are equivalent (0.784 & 0.773 for the phoneme dataset and 0.889 & 0.882 for the Forest cover datasets).

Table 4-19. Comparison of Hand-picked Parameters with Under-sample SMOTE Algorithm Parameters (C4.5 as base classifier)

Datasets	Under-sample SMOTE Algorithm picked parameters (Mode)		Hand-picked parameters [7]
	SMOTE percentage	Under-sampling percentage	SMOTE percentage
Phoneme	100%	100%	100%
Satimage	500%	100%	200%
Mammography	100%	90%	400%
Forest cover	300%	100%	300%
Pima Indian	200%	100%	400%
Oil	300%	100%	500%

Table 4-20. Comparison of Results between Hand-picked Parameters with Under-sample SMOTE Algorithm Picked Parameters (C4.5 as base classifier)

Datasets	Results using Wrapper selected SMOTE & under-sampling percentages (Mode)			Results using hand-picked SMOTE & under-sampling percentages [7]		
	Average Minority class TP-rate	Average Minority class F-value	Average Majority class F-value	Average Minority class TP-rate	Average Minority class F-value	Average Majority class F-value
Phoneme	0.849	0.784	0.899	0.784	0.773	0.904
Satimage	0.639	0.570	0.947	0.519	0.542	0.953
Mammography	0.626	0.646	0.992	0.512	0.616	0.992
Forest cover	0.904	0.889	0.991	0.868	0.882	0.991
Pima Indian	0.773	0.653	0.755	0.869	0.656	0.704
Oil	0.448	0.424	0.972	0.647	0.444	0.962

For the Satimage dataset, our wrapper algorithm selected a much higher SMOTE percentage (500%) compared to the hand-picked SMOTE percentage (200%), and surprisingly our minority class TP-rates and F-values were better than those obtained by hand-picked method. For the Mammography dataset, our results were better than hand-picked results when our algorithm selected a lesser SMOTE percentage as compared to that selected by Chawla et al. [7]. It was for the Oil dataset, where our results were significantly lower than those obtained in [7] which selected 500% SMOTE for minority class. On the whole our wrapper algorithm had selected SMOTE percentage which gave comparable results to those by Chawla et al. [7]. Results in [7] should have been always better than ours as they were hand-picked, but in some cases where lower than our results. The reason for this might be because of the different performance measure Chawla et al. [7] were trying to optimize.

4.8. Results Using a Brute Force Search Method

To cross check whether our heuristic search using under-sampling followed by SMOTE was valid in finding a good under-sampling and SMOTE mix for majority and minority classes, we searched the parameter space using a Brute Force Search Method, where a set of discrete valued combinations of under-sampling and SMOTE percentages for majority and minority classes are tested. Since this process is extremely time consuming, the experimental runs were performed using the C4.5 learning algorithm only on two relatively small dissimilar datasets: the Phoneme and Mammography datasets. The SMOTE percentages were varied from 0% (no SMOTE) to 1000% in steps of 100% for the minority class and under-sampling percentages from 100% to 50% in steps of 10% for the majority class. So in all 11 (SMOTE) x 6 (Under-sampling) = 66 combinations of SMOTE and under-sampling percentages were tested. For each of the 10 cross-validation folds, a five fold cross-validation wrapper search using the Brute Force method was performed using all of the 66 under-sampling and SMOTE percentage combinations. Note that the search is done using a wrapper based algorithm using Brute Force Method instead of using any heuristic. The under-sampling and SMOTE percentage combination which gave the highest F-value for the minority class on the five fold cross-validation was used on the whole training data and then evaluated on the test data. So a total of 10 sets of under-sampling and SMOTE percentages and corresponding wrapper performance metrics were obtained for each of the 10 cross-validation folds which were then statistically compared with wrapper results obtained using our heuristic search. The baseline results for both search methods are identical as the same training/testing set pairs were used.

Table 4-21 shows the results obtained using the brute force method and how they compare with the results obtained by search heuristic of Under-sample SMOTE Algorithm. One can see that the brute force method selects much higher percentages for under-sampling (50% and 54% average under-sampling percentage over 10 folds for the Phoneme and Mammography datasets) and SMOTE and gets better accuracies on the five-fold cross-validation training data as compared with the wrapper based algorithm, but it is not effective in boosting accuracy over the testing data.

Table 4-21. Results Using Brute Force Method

		Phoneme	Mammography
Average SMOTE percentage	Wrapper Brute Force	250%	190%
	Wrapper Under-Sample SMOTE Algorithm	150%	180%
Average under-sampling percentage	Wrapper Brute Force	50%	54%
	Wrapper Under-Sample SMOTE Algorithm	89%	87%
Average Minority class TP-rate	Wrapper Brute Force	0.912	0.652
	Wrapper Under-Sample SMOTE Algorithm	0.875	0.659
	% increase	-4.26%	1.17%
	t-stat	3.310	-0.252
	Significance	√ (worse)	-
Average Minority class F-value	Wrapper Brute Force	0.754	0.619
	Wrapper Under-Sample SMOTE Algorithm	0.780	0.634
	% increase	3.31%	2.42%
	t-stat	-5.367	-0.832
	significance	√	-
Average Majority class F-value	Wrapper Brute Force	0.863	0.990
	Wrapper Under-Sample SMOTE Algorithm	0.891	0.991
	% increase	3.14%	0.07%
	t-stat	-5.893	-0.751
	significance	√	-

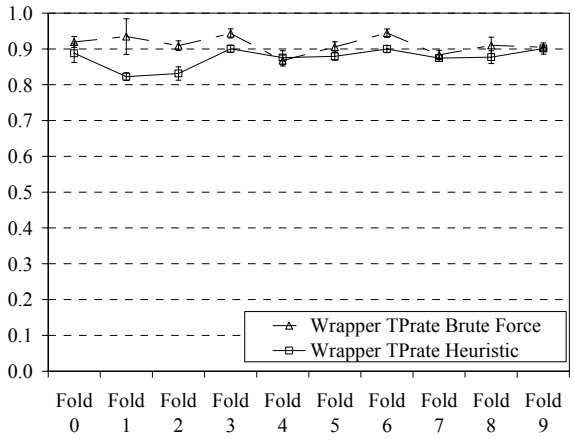

Figure 4-49. Comparison of Brute Force and Heuristic Search for Wrapper True Positive Rate on Phoneme Data

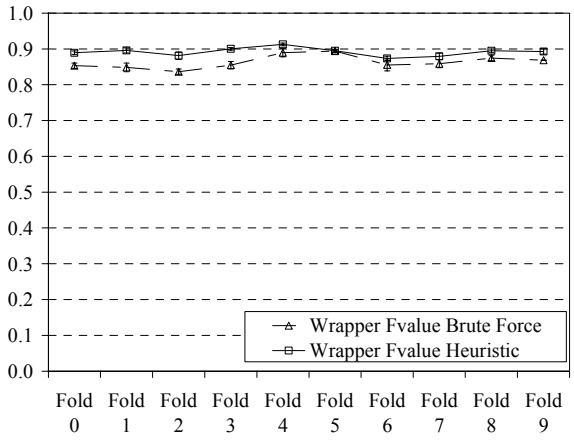

Figure 4-50. Comparison of Brute Force and Heuristic Search for Wrapper F-value on Phoneme Data

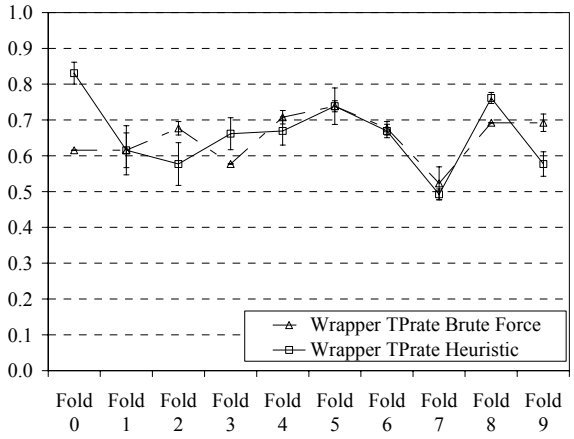

Figure 4-51. Comparison of Brute Force and Heuristic Search for Wrapper True Positive Rate on Mammography Data

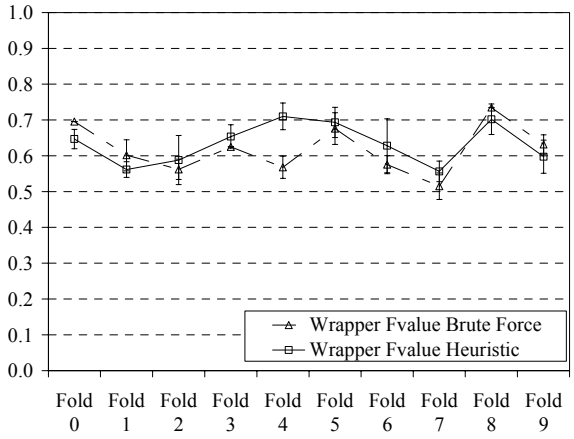

Figure 4-52. Comparison of Brute Force and Heuristic Search for Wrapper F-value on Mammography Data

From Table 4-21, Figure 4-51 and Figure 4-52 for the Phoneme dataset which had a smaller imbalance, one can see that the brute force method's TP-rates were statistically significantly better than the TP-rates obtained by the Under-sample SMOTE algorithm, but also had a statistically significant reduction in F-values over both the classes which was not desirable. From Table 4-21, Figure 4-49 and Figure 4-50 for the Mammography dataset, one can note that there was no statistical difference between the two result sets for all the measures, although the Under-sample SMOTE algorithm performed better than the brute force method by a getting higher average TP-rate over the minority class and average F-values over both the classes.

Thus this seems to support our hypothesis about the heuristic of giving lesser importance to under-sampling over the SMOTE, which also happens to conform with the theory that removing data may result in reduction in the amount of information and less accuracy, while the addition of new noise free data will always add more information leading to better accuracy.

CHAPTER 5

CONCLUSION & FUTURE WORK

From previous work [7], SMOTE was found to work really well in expanding the decision regions of classifiers for minority classes and increase the classifiers performance over minority examples on the unseen test data. However, the percentage of SMOTE performed was an important parameter as a smaller percentage of synthetic examples could result in less recall and a large percentage of synthetic examples could result in less precision, both of which were not desired. So in our study we have used the F-value as a performance metric for guiding the search process which incorporates both measures - Recall which gives us the measure of completeness of the classifier in predicting the actual minority class and Precision which gives us the measure of correctness of the classifier in predicting the actual minority class. Our wrapper Under-sample SMOTE algorithm performs five-fold cross-validation and uses the averaged F-value on five validation sets as representative of the F-value on the test data to guide the search through the parameter space of under-sampling and SMOTE percentages. The results show that our algorithm is able to successfully select the under-sampling percentages for the majority classes and SMOTE percentages for the minority classes for specific learning algorithm, which when used on the whole training data significantly improves the recall for the minority classes on the test data with a small amount of reduction in the F-value over the majority classes. The results also show that there is no clear benefit in under-sampling the majority classes as SMOTE alone can achieve similar results. The Brute Force results which perform heavy under-sampling may improve recall on minority classes at the expense of reduced F-values (higher false positives) of minority and majority classes which is not desirable and thus undermines the importance of under-sampling majority classes.

For future work, more diverse machine learning algorithms should be used to evaluate our Under-sample SMOTE algorithm. In a recent study by Batista et al [61] , they have experimented with various

sampling techniques and have found SMOTE plus focused under-sampling using Tomek links [23] and Wilson's Edited Nearest Neighbor Rule [62] give very good results. Since random under-sampling is found to have no apparent accuracy benefit, future work should use these focused under-sampling techniques along with SMOTE to see if there is any significant improvement in results compared to results in this study. Guo and Viktor [63] have found that giving more importance to the hard to classify examples from both minority and majority classes for creating synthetic examples is advantageous. Future work should use this information to extend SMOTE to 'hard to classify examples' from both classes. Also the parameter k – number of nearest neighbors used in SMOTE algorithm could happen to be an important parameter and more work should to done to confirm this hypothesis and if true it should be fine tuned automatically for each dataset and specific learning algorithm using the wrapper approach.

REFERENCES

[1] Nua online internet surveys portal for market research, statistics, demographics, metrics, surveys and web trends from across the globe. http://www.nua.com/surveys/index.cgi.

[2] W. Lee, S. J. Stolfo, Data Mining Approaches for Intrusion Detection, *Proceedings of the 1998 USENIX Security Symposium*, 1998.

[3] E. Bloedorn, et al., Data Mining for Network Intrusion Detection: How to Get Started, *MITRE Technical Report*, August 2001.

[4] J. Luo, Integrating Fuzzy Logic With Data Mining Methods for Intrusion Detection, *Master's thesis, Department of Computer Science, Mississippi State University*, 1999.

[5] D. Barbara, N. Wu, S. Jajodia, Detecting Novel Network Intrusions Using Bayes Estimators, *First SIAM Conference on Data Mining*, Chicago, IL, 2001.

[6] A. Lazarevic, L. Ertoz, A. Ozgur, J. Srivastava, V. Kumar, A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection*, Proceedings of Third SIAM Conference on Data Mining,* San Francisco, May. 2003.

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

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

[9] N. Chawla, A. Lazarevic, L. Hall, K. Bowyer, SMOTEBoost: Improving Prediction of Minority Class in Boosting*, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Cavtat-Dubrovnik, Croatia, pp. 107-119, 2003.

[10] F. Provost and T. Fawcett, Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions, In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pp. 43-48, 1997.

[11] T. Fawcett and F. Provost, Combining Data Mining and Machine Learning for Effective User Profile, In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 8-13, Portland, OR, 1996. AAAI.

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

[13] D. Lewis and J. Catlett. Heterogeneous Uncertainty Sampling for Supervised Learning. In *Proceedings of the Eleventh International Conference of Machine Learning*, pp. 148-156, 1994.

[14] M. Kubat, R. Holte, and S. Matwin. Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning*, Vol. 3, pp. 195-215, 1998.

[15] In N. Japkowicz, editor, *Proceedings of the AAAI'2000 Workshop on Learning from Imbalanced Data Sets, AAAI Tech Report WS-00-05.* AAAI 2000.

[16] In N.V. Chawla, N. Japkowicz, and A. Kolcz, editors, *Proceedings of the IMCL'2003 Workshop on Learning from Imbalanced Data Sets,* ICML 2003.

[17] In N.V. Chawla, N. Japkowicz, A. Kolcz, S. Sarawagi and U. Fayyad, editors, *ACM SIGKDD Exploration Newsletter,* Volume 6(1), 2004.

[18] P. Domingos. MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 155-164, San Diego, CA, 1999. ACM.

[19] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, C. Brunk. Reducing Misclassification Costs. In *Proceedings of the Eleventh International Conference on Machine Learning*, 1994.

[20] M. Kubat, and S. Matwin. Addressing the Curse of Imbalanced Training Sets: One Sided Selection. In *Proceedings of the Fourth International Conference on Machine Learning*, pp. 179-186, 1997.

[21] D.X. Ling and C. Li. Data Mining for Direct Marketing: Problems and Solutions. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), 1998.

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

[23] I. Tomek. Two modifications of CNN. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 6, pp. 769-772, 1996.

[24] K. M. Ting. An instance-weighted method to induce cost-sensitive trees. *IEEE Transaction on Knowledge and Data Engineering*, 14:659-665, 2002.

[25] P. Turney. Types of cost in inductive concept learning. In *Proceedings of the ICML'2000 Workshop on Cost-Sensitive Learning*. Pages 15-21, 2000.

[26] D. Tax, *One Class Classification,* PhD Thesis, Delft University of Technology, 2001.

[27] L.M. Manevitz and M. Yousef, One-class SVMs for document classification. *Journal of Machine Learning,* pp. 2:139-154, 2001.

[28] N. Japkowicz, C. Myers and M. Gluck. A novelty detection approach to classification. In *Proceedings of the Fourteenth IJCAI Conference*, Montreal, pp. 518-523, 1995.

[29] M. Thien and H. Bunke. Off-line, Handwritten Numeral Recognition by Perturbation method. *Pattern Analysis and Machine Intelligence*. Vol. 19/5, pp. 535-539, 1997.

[30] R. Kohavi and G. John. The wrapper approach. In *Feature Selection for Knowledge Discovery and Data Mining*, H. Liu and H. Motoda (eds.), Kluwer Academic Publishers, pp. 33-50, 1998.

[31] P. Langley and S. Sage. Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399-406, Seattle, WA. 1994. Morgan Kaufmann.

[32] M. Pazzani. Searching for attribute dependencies in Bayesian classifiers. In Fisher, D. and Lenz, H., editors. *Fifth International Workshop on Artificial Intelligence and Statistics*, pages 424-429, Ft. Lauderdale, FL.

[33] M. Singh and G.M. Provan. A comparison of induction algorithms for selective and non-selective Bayesian classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 497-505, Lake Tahoe, CA, 1995. Morgan Kaufmann.

[34] R. Kohavi and G. John. Wrappers for Feature Subset Selection. *Artificial Intelligence journal*, Vol. 97, Nos. 1-2, pp. 273-324, 1997.

[35] R. Kohavi and G. John. Automatic Parameter Selection by Minimizing Estimated Error. In Prieditis, A. and Russell, S., editors. In *Machine Learning: Proceedings of the Twelfth International Conference*, pp. 304-312, 1995. Morgan Kaufmann.

[36] D. B. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 293-301, 1994. Morgan Kaufmann.

[37] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203-231,2001.

[38] A. P. Bradley. The use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms. *Pattern Recognition*. Vol. 30(6), pp. 1145-1159, 1997.

[39] J. Swets. Measuring the Accuracy of Diagnostic System. In *Science*. pp. 1285-1293, 1988.

[40] M. Buckland and F. Gey. The relationship between recall and precision. In *Journal of the American Society for Information Science*. 45(1) pp. 12-19, 1994.

[41] M. Joshi, V. Kumar, R. Agarwal. Evaluating Boosting Algorithms to Classify Rare Classes: Comparison and Improvements. *First IEEE International Conference on Data Mining*. San Jose, CA, 2001.

[42] J. R. Quinlan. C4.5: Programs for Machine Learning. 1993. Morgan Kaufmann.

[43] F. Provost, T. Fawcett, and R. Kohavi. The Case Against Accuracy Estimation for Comparing Induction Algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 445-453, Madison, WI, 1998.

[44] ELENA Project. ftp.dice.ucl.ac.be in the directory of /pub/neural-nets/ELENA/database.

[45] C. Blake and C. Merz, UCI Repository of Machine Learning Databases. http://www.ics.uci.edu/~mlearn/MLRepository.html, Department of Information and Computer Science, University of California, Irvine, 1998.

[46] S. Cost and S. Salzberg. A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning*. vol. 10, no. 1, pp. 57-78, 1993.

[47] William W. Cohen. Fast effective rule induction. In Machine Learning: *Proceedings of the Twelfth International Conference*, Lake Tahoe, California, 1995.

[48] J. Ross Quinlan. Learning efficient classification procedures and their application to chess end games. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning - An Artificial Intelligence Approach*, pp. 463-482. Tioga, Palo Alto, CA, 1983.

[49] Johannes Furnkranz and Gerhard Widmer. Incremental reduced error pruning. In *Machine Learning: Proceedings of the Eleventh Annual Conference*, New Brunswick, New Jersey, 1994. Morgan Kaufmann.

[50] Johannes Furnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review* 13(1): 3-54, 1999.

[51] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning,*5(3), 1990.

[52] J. Rissanen. Stochastic complexity and modeling. *Ann. Statist.,* Vol. 14, pp. 1080-1100, 1986.

[53] D. Hull. Using Statistical Testing in the Evaluation of Retrieval Experiments. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329-338. ACM Press, 1993.

[54] E. Michael Keen. Presenting results of experimental retrieval comparisons. *Information Processing & Management*, Vol. 28, No. 4, pp. 491-502, 1992.

[55] KDD-cup 1999 Task Description, http://kdd.ics.uci.edu/databases/kddcup99/task.html.

[56] Richard P. Lippmann, David J. Fried, Isaac Graf, Joshua W. Haines, Kristopher R. Kendall, David McClung, Dan Weber, Seth E. Webster, Dan Wyschogrod, Robert K. Cunningham, and Marc A. Zissman, Evaluating Intrusion Detection Systems: the 1998 DARPA Off-Line Intrusion Detection Evaluation, *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, Vol. 2, pp. 12-26, IEEE Computer Society Press, Los Alamitos, CA, 2000.

[57] Defense Advanced Research Project Agency. DARPA Intrusion Detection Evaluation, http://www.ll.mit.edu/IST/ideval/index.html.

[58] C. Elkan. Results of the KDD'99 classifier learning contest. In http://www-cse.ucsd.edu/~elkan/clresults.html, September 1999.

[59] Nahla Ben Amor, Salem Benferhat, Zied Elouedi: Naive Bayes vs decision trees in intrusion detection systems. *Proceedings of the 2004 ACM symposium on Applied computing*, 420-424.

[60] M. R. Sabhnani, and G. Serpen, Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context, *Proceedings of International Conference on Machine Learning: Models, Technologies, and Applications*, Las Vegas, Nevada (2003), 209-215.

[61] Gustavo E. A. P. A. Batista and Ronaldo C. Prati and Maria Carolina Monard, A study of the behavior of several methods for balancing machine learning training data *ACM SIGKDD Exploration Newsletter,* Volume 6(1), pp. 20-29, 2004.

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

[63] H. Guo and H. L. Viktor. Learning from imbalanced data sets with boosting and data generation: The DataBoost-IM approach. *SIGKDD Explorations, ACM SIGKDD Exploration Newsletter,* Volume 6(1), pp. 30-39, 2004.