The Effect of Oversampling and Undersampling on Classifying Imbalanced Text Datasets

by

Alexander Yun-chung Liu, B.S.

Thesis

Presented to the Faculty of the Graduate School
of The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of
Master of Science in Engineering

The University of Texas at Austin
August 2004

The Effect of Oversampling and Undersampling on
Classifying Imbalanced Text Datasets

APPROVED BY
SUPERVISING COMMITTEE:

Acknowledgements

This document could not have been finished without the help and contributions of several important people. First and foremost, I would like to thank my supervising professor Dr. Joydeep Ghosh, not only for his suggestions and guidance on this paper, but also for his advice on being a better graduate student and contributing member of society in general. I would also like to thank Sara Matzner for struggling through rough copies of this thesis and offering helpful suggestions, along with her stories and reminders of how to be a focused graduate student. Thanks go to Curtis Shrote as well, who was originally slated to be a reader, but has gone on to do greater, better things. These are, unfortunately, greater, better, non-university related things, which invalidated Curtis as reader.

I must also thank many others who have helped form the ideas presented here. All faults and errors are my own, but everything else can be traced back to helpful discussions with everyone in LANS (Sreangsu Acharyya, Arindam Banerjee, Gunjan Gupta, Chase Krumpelman, Srujana Merugu, Kunal Punera, and Suju Rajan), the group at CIADS (Michael Anderson, Donna Bustamantes, Tom Henderson, Bernie Lofaso, and Daniel McCabe), who not only put up with my hogging of processor time and memory, but also helped me foot the bill, and, of course, all the researchers whose work I have cited in this document. Throughout my graduate and undergraduate studies, I was also helped financially by many generous donors, including a Texas Exes Award for Scholarship and Leadership, a Thrust Fellowship, and a Basdall Gardner Memorial MCD Fellowship in Engineering. I would be remiss without thanking Eleanor Moore, Melanie Gulick, Dean Neikirk, Michael Feissli, Patsy Megannis, and Dottie Beatty for all of their help with the various paperwork and regulations needed to submit a thesis.

In the space I have left to thank people, I would like to issue a general blanket statement of thanks to all those who have known me in the past but whose names are too long to fit on a single page. Specifically, however, I would like to thank my family for all of their support throughout the years and without whom I would literally not be here today. In particular, I thank my mom and dad for all of their sage advice and support and my sister and brother-in-law for constantly asking about my graduation date and threatening to buy plane tickets to come visit me at graduation, forcing me to make sure there would actually be a graduation when they arrived. Finally, thank God for all that He has done and all that He will do and for allowing me to type the last period that will go into this thesis right here.

The Effect of Oversampling and Undersampling on
Classifying Imbalanced Text Datasets

by

Alexander Yun-chung Liu, M.S.E.
The University of Texas at Austin, 2004
SUPERVISOR: Joydeep Ghosh

Many machine learning classification algorithms assume that the target classes share similar prior probabilities and misclassification costs. However, this is often not the case in the real world. The problem of classification when one class has a much lower prior probability in the training set is called the imbalanced dataset problem. One popular approach to solving the imbalanced dataset problem is to resample the training set. However, few studies in the past have considered resampling algorithms on data sets with high dimensionality. In this thesis, we examine the imbalanced dataset problem in the realm of text classification. Text has the added problems of both sparsity and high dimensionality. We first describe the resampling techniques we use in this thesis, including several resampling techniques we are introducing. After resampling, we classify the data using multinomial naïve Bayes, k nearest neighbor, and SVMs. Finally, we compare the results of our experiments and find that, while the best resampling technique to use is often dataset dependent, certain resampling techniques tend to perform consistently when coupled with certain classifiers.

Table of Contents

1.0 Introduction

Many traditional approaches to machine learning classification problems assume that the target classes share similar prior probabilities. However, in many real world problems, this assumption is grossly violated. Often, it is the case that the ratios of prior probabilities between classes are extremely skewed. This situation is known as the imbalanced dataset problem. Several important problems involving imbalanced datasets include mammography, oil-spill detection, network intrusion detection, fraud detection, and several other types of anomaly detection. The amount of imbalance varies depending on the problem. In intrusion detection, it is not uncommon for less than ten percent of the records to represent actual intrusions. In detection of cancerous cells, typically less than one percent of cells are actually cancerous.

We will refer to the class with the smallest prior probability as the target or minority class throughout the rest of this thesis, with the assumption that there is only one minority class of interest. In the imbalanced dataset problem, it is typically the case that it is more important to correctly classify the minority class. For example, in the case of mammography, it is much more important to correctly detect cancerous cells. While misclassifying non-cancerous cells leads to additional testing (and an eventual clean bill of health), misclassifying cancerous cells leads to serious health risks. In this and many other settings involving imbalanced datasets, the cost of incorrectly classifying the minority class is often much higher than the cost of incorrectly classifying the majority, or non-target class.

Most studies of the imbalanced dataset problem have dealt with low dimensional data. However, several real world problems involve extremely high dimensionality. As one increases the dimensionality of the feature space, the feature space naturally becomes less densely populated. One particularly important realm of machine learning which must deal with extremely high dimensional spaces is the realm of text datasets. When dealing with text, the most popular representation is to represent each document as a vector of word frequencies. That is, each word in the vocabulary becomes a feature. A modestly sized document collection may have a vocabulary in the tens of thousands, leading to feature vectors which are both high in dimensionality and extremely sparse.

Few papers in the past have addressed the problem of text classification in conjunction with the imbalanced dataset problem. Of those papers that do study text in an imbalanced dataset scenario, to our knowledge only one paper uses resampling techniques to address the imbalanced dataset problem [Nickerson01]. Moreover, [Nickerson01] only studied the effect of resampling

on a single dataset and a single classifier. Thus, there is a need for a study comparing the effect of resampling techniques on several text datasets and on several classification algorithms.

In this thesis, we study the imbalanced dataset in the context of text classification. Text classification broadly refers to the task of separating documents into predefined topical categories based on document content. Several text classification problems involve imbalanced datasets. For example, when retrieving documents based on user queries, typically very few documents will match the user's parameters. This is particularly true when retrieving documents on the Internet. Thus, it is important to address the imbalanced dataset problem in the realm of text classification. As in other domains, we will assume that it is much more costly to misclassify the minority class.

The rest of this thesis is organized as follows. In section 2, we will describe previous approaches to the imbalanced dataset problem and further justify our approach to this problem. In section 3, we will describe the resampling techniques we will be studying in this thesis. In section 4, we will describe our experimental setup, along with the datasets, classifiers, and evaluation metrics which will be used in our empirical evaluation. In section 5, we will describe the results of our experiments. Finally, in section 6, we will describe possible areas of future work and conclude our thesis.

2.0 Previous work

There are generally two methods of dealing with imbalanced datasets [Japkowicz04]. The first method is to modify the classifier. Several studies have examined the use of cost sensitive learning. For example, in [Brank03], the authors study the effect of cost sensitive SVMs on text classification. In [Wu03], the authors modify the SVM classifier itself in order to handle imbalance datasets. Another approach taken is to change the problem into a one-class classification problem. Here, the classifier is trained on the majority class, and the minority class is identified by detecting anomalies.

The second general approach to solving the imbalanced dataset problem is to modify the data itself. The most widespread and typical way to modify the dataset is to resample the data. Resampling refers to the process of changing the prior probabilities of the majority and minority class in the training set by changing the number of records in the majority and minority class. In particular, oversampling refers to the process of increasing the number of records in the minority class while undersampling refers to the process of decreasing the number of records in the majority class. In general, resampling has several advantages over modifying the classification algorithm and cost sensitive learning. As stated in [Japkowicz04], a modified classification algorithm may only be useful in only a few domains if the classifier becomes too heuristic to be applied in other domains. Since imbalanced datasets are found in so many different domains of application, it is highly desirable to find a more universal approach to the problem. In addition, many standard classification algorithms and programs are already available or might already be in use by organizations. It is much easier to change the data that goes into a standard classification algorithm then to replace or change an already intact system. Finally, in the realm of cost sensitive learning, it may be hard to quantify costs. For example, in the intrusion detection setting, how much more does a missed intrusion cost compared to a caught intrusion? How much more costly is it to misclassify cancerous cells rather than to correctly diagnose them? These costs are often difficult to quantify, even if one knows that one class is much more important than others. Moreover, it may be possible to achieve the same results via resampling. In [Maloof03], the authors study the relationship between cost sensitive learning and resampling. Interestingly, for a naïve Bayes classifier, the authors were able to achieve almost the same results via resampling and via cost sensitive learning. However, the authors state that the work is still preliminary, meaning that further investigation is needed to determine the exact relationship between cost sensitive learning and resampling methods. Thus, resampling is a simple and attractive alternative to modifying the classification algorithm and cost sensitive learning.

In the past, there have been several studies on text classification in an imbalanced dataset scenario, though only [Nickerson01] considers resampling. One particularly interesting study is [Zheng03] and [Zheng04]. In both papers, the authors study the effect of feature selection on text classification. In particular, the authors show that choosing a combination of words that are highly indicative of class membership and words that are not indicative of class membership yields better performance than standard feature selection methods. In [Serrano04], the authors also try to handle imbalanced text classification by using genetic algorithms to help guide feature selection. However, we do not follow either method of feature selection, instead relying on traditional feature selection in the text classification domain. We justify this decision when we describe our experimental setup.

Most studies on text and imbalanced datasets, however, only empirically study one dataset and few classifiers. For example, both [Zheng04] and [Zheng04] studies only the Reuters-21578 dataset and multinomial naïve Bayes. [Brank03] studies the documents from the Reuters corpus, volume 1 and SVMs using cost sensitive learning. [Nickerson01] studies the Reuters-21578 dataset and decision trees after attempting to account for both inter-class and intra-class imbalance via resampling with very poor results. [Serrano04] studies three datasets, the Reuters-21578 dataset, a personally collected dataset, and a dataset based on Yahoo hierarchies. However, the authors do not consider resampling and only use naïve Bayes.

Finally, [Japkowicz03] and [Jo04] indicate that the imbalanced dataset problem may actually arise from two different problems. The first is the problem of interclass imbalance, which we have already described. Interclass imbalance refers to the very different prior probabilities that each class may have. The second problem is within class (also called intraclass) imbalance. Within class imbalance may occur when the members of a class are not distributed in a unimodal distribution. That is, members from one distribution in a particular class are underrepresented when compared to other members of the same class drawn from different distributions. When resampling techniques are applied to fix the imbalanced dataset problem, within class imbalance is often ignored. These resampling techniques may actually worsen within class imbalance [Japkowicz02].

One solution to the imbalanced dataset problem that addresses both interclass and within class imbalance is described in [Nickerson01]. In this solution, the authors first cluster the minority class and the majority class separately. The authors then examine the cluster memberships and resample based on the number of items per cluster instead of the number of

items per class as with most resampling techniques. In this way, intraclass imbalance is addressed along with interclass imbalance.

We must now address the issue of how our study handles the within class imbalance problem. Unfortunately, the resampling techniques and even our experimental setup largely ignore the within class imbalance problem. Our study shares several flaws with several previous studies on resampling methods. Many resampling methods do not address the within class imbalance problem. Instead, they seek to address different aspects of class distribution when choosing which records to resample. With regards to experimental setup, we have followed the lead of several past studies in order to produce comparable results. Typically, when studying the class imbalance problem, researchers have posed the problem as a two class problem. However, most of the datasets used are multiclass datasets. Thus, in order to study the problem as a two class dataset, several classes are combined together. A common approach is to take the smallest of the original classes as the minority class and combine the remaining classes to form the majority class. When doing this, however, we create a multimodal majority class. This results in some degree of within class imbalance in the majority class.

Thus, we must take a moment to justify why we are comparing the effects of different resampling methods without taking into account the within class imbalance problem. Given the approach in [Nickerson01] to solving the within class imbalance problem, at some point, the clusters need to be resampled. In their original work, the authors use random oversampling and random undersampling. However, we could potentially use a variety of resampling methods at this point. By first studying the interclass imbalance problem, we can use these techniques to address the intraclass imbalance problem. Finally, by first obtaining results on the inter class imbalance problem, we can eventually compare these results with the results obtained for solving both types of imbalance problems. Since solving both interclass and intraclass imbalance is significantly more complex than solving just the interclass imbalance problem, we would need to justify (via significantly improved results) the overhead for solving both problems. We leave this comparison to future work.

However, before departing the problem of within class imbalance, we end with some final questions that need to be addressed in future work. The first problem is that of distinguishing rare occurrences within the minority class with noise within the minority class. By definition, we have few members of the minority class. If there are records that rarely occur within the minority class, we will have extremely few occurrences of these records. In particular, if these members are extremely similar to members of the majority class, we would want to

emphasize these members if they are truly part of the minority class, but completely ignore these members if they are simply noise. The second problem is that of distinguishing rare occurrences within the majority class. Typically, in the imbalanced dataset problem, it is more costly to misclassify members of the minority class than members of the majority class. Thus, if there are rare members of the majority class similar to the minority class, it may not be worth the extra overhead to weed out these false positives, particularly since they are so rare. The decision to deal with both interclass and intraclass imbalance is most likely domain and problem dependent.

3.0 Resampling techniques

In the following section, we will describe the resampling methods used in this experiment. Let us first note that this is not meant to be an exhaustive list of published resampling methods. For example, there are several important resampling methods used in other studies (e.g. [Kubat97], [Batista04], [Barandela03]) that we do not include in our experiments. Instead, we have chosen either those resampling techniques that have performed well in the past or those resampling techniques which are representative of other resampling techniques in terms of how data points are selected for resampling.

Most studies on resampling focus on undersampling [Zheng03][Barandela03]. Thus, we introduce several new oversampling techniques. Since several past studies have shown what types of approaches to undersampling are promising, we study different oversampling techniques simply to see if any of these new techniques are promising on high dimensional datasets. In addition, all of our resampling techniques allow us to resample until we reach a desired ratio between the minority and majority class, allowing us to directly compare different resampling methods for a given ratio of minority and majority class datapoints in the final training set. Finally, we do not consider schemes where both the minority class is oversampled and the majority class is undersampled simultaneously. While such an approach could improve the results in this study, we do not consider any such schema since we want to better understand the effects of undersampling and oversampling in isolation.

Finally, it is interesting to point out that all of the studies we are aware of (including our own) never eliminate members from the minority class. Since there are so few members of the minority class, researchers are very hesitant to eliminate members of the minority class. Instead, the assumption is that each minority class member is very important. This may or may not be the case in practice since some minority class members may represent noise. However, with so few data points, it is difficult to differentiate between noise and minority class members which represent very rarely seen documents (i.e. class disjuncts in the intra class imbalance problem).

3.1 Similarity metric

Several of the resampling techniques described below select records based on similarity metrics. In this study, we use the cosine similarity as our similarity metric. Formally, for documents d_i and d_j, the cosine similarity between d_i and d_j is given by:

$$cossim(d_i, d_j) = \frac{< d_i, d_j >}{||d_i||||d_j||}$$

This is the standard definition for cosine similarity (i.e. the normalized inner product). We choose this metric because the cosine similarity has been shown to work well in the realm of text classification. All similarity measures in our study, particularly those used to find nearest neighbors, use the cosine similarity.

3.2 Undersampling techniques

Undersampling seeks to reduce the number of majority class members in the training set [Kubat97]. As a result, the overall number of records in the training set is greatly reduced. This means that during classification, training time is also greatly reduced. Since we are dealing with very high dimensional datasets, there is a significant savings in memory as well. However, because we are eliminating members from the majority class, it is possible that we will lose a lot of valuable information if we eliminate documents that could be useful to our classifier in building an accurate model.

3.2.1 Random undersampling

Random undersampling is a simple approach to resampling. Majority class documents in the training set are randomly eliminated until the ratio between the minority and majority class is at the desired level. Theoretically, one of the problems with random undersampling is that one cannot control what information about the majority class is thrown away. In particular, very important information about the decision boundary between the minority and majority class may be eliminated. Despite its simplicity, random undersampling has empirically been shown to be one of the most effective resampling methods. In particular, few of the more sophisticated undersampling methods have outperformed random undersampling in empirical studies.

3.2.2 Undersampling techniques from [Zhang03]

The following four undersampling methods are taken from [Zhang03] with some adjustment in order to be suited for high dimensional datasets (i.e. choice of distance metric). We refer to the undersampling methods with the names given to them in [Zhang03], except for the method we call "Distant1." In the original paper, this method was simply called "Distant," but we have renamed it in order to differentiate this approach from two new approaches named "Distant2" and "Distant3." In addition, the NearMiss methods share a relationship with their corresponding Distant method (e.g. Nearmiss1 shares a relationship with Distant1), a fact that is more easily remembered with this naming scheme.

The four resampling methods introduced in [Zhang03] are NearMiss1, NearMiss2, NearMiss3, and Distant1. In NearMiss1, we select the majority class members whose average distance to the three closest minority class members in the training set is smallest. In NearMiss2, we select the majority class members whose average distance to the three farthest minority class members in the training set is smallest. In NearMiss3, for each minority class member in the training set, we select the n closest majority class documents in the training set, where n is chosen based on the desired ratio between minority and majority class documents in the final, undersampled training set.

We must note that Zhang03 leaves out some details of the NearMiss3 approach. In particular, it is unclear in NearMiss3 what happens when a document from the majority class that has already been selected to be in the training set also happens to be near another document from the minority class. We interpret NearMiss3 as follows: We have a target number of documents to be in the undersampled majority class. For each document in the minority class, we find the closest n documents from the majority class, where n=(target num majority class documents) / (number minority class documents). It is possible for a document from the original training set to be included more than once in the final training set. This may or may not be the implementation of NearMiss3 described in [Zhang03], where it seems that they might have a scheme that avoids duplication in the undersampled training set.

The intuition behind the NearMiss methods, as described in [Zhang03], is as follows. In NearMiss1, we select majority class documents that are very similar to some minority class documents. In NearMiss2, we select majority class documents that are very similar to all of the minority class documents. In NearMiss3, we select majority class documents in a way that we guarantee the minority class documents are surrounded by majority class documents. Thus, the NearMiss methods can be thought of as three different ways of trying to better isolate the decision boundary between the majority and minority class. Those majority class documents that are far from the decision boundary are eliminated from the undersampled training set.

Finally, let us describe the fourth undersampling method in [Zhang03]. In Distant1, we select the majority class documents whose average distance to the three closest minority class documents is largest. We will discuss Distant1 more thoroughly after introducing Distant2 and Distant3, variants on techniques from [Zhang03].

3.2.3 Variations on techniques from [Zhang03]

The following two undersampling methods are based on the work in [Zhang03]. We call these two methods Distant2 and Distant3. In Distant2, we select the majority class members whose average distance to the three farthest minority class documents is largest. In Distant3, for each minority class member in the training set, we select the n farthest majority class documents in the training set, where n is chosen based on the desired ratio between minority and majority class documents in the final, undersampled training set. Thus, the Distant methods try to eliminate majority class documents that are similar to minority class documents. Thus, as expected in [Zhang03], we will most likely achieve higher recall at the cost of precision with the Distant methods, and higher precision at the cost of recall with the NearMiss methods.

In particular, note that NearMiss1 and Distant1 are, in a way, analogous to each other. That is, NearMiss1 chooses documents whose average distance to its three closest documents is smallest, while Distant1 chooses documents whose average distance to its three closest documents is largest. Distant2 and Distant3 were created to share a similar relationship to NearMiss2 and NearMiss3, respectively. Finally, we should note that these undersampling methods were used in a study on kNN used on low-dimensional data [Zhang03].

3.3 Oversampling techniques

Oversampling seeks to increase the number of minority class members in the training set. The advantage of oversampling is that no information from the original training set is lost since we keep all members from the minority and majority classes. However, the disadvantage is that we greatly increase the size of the training set. Thus, we also increase training time and the amount of memory required to hold the training set. Since we are dealing with very high dimensional datasets, we need to be careful how we proceed in order to keep time complexity and memory complexity under reasonable constraints. If we do not consider the time taken to resample, undersampling beats oversampling in terms of time and memory complexity. Since this is the case, oversampling needs to be significantly better than undersampling in terms of classification performance in order to be viable. Past studies have not reached any conclusive results with regards to whether undersampling or oversampling is best with regards to classification performance. Most likely, conflicting results are due to the combination of different datasets and classification algorithms. In addition, choice of resampling method is probably both domain and problem specific.

3.3.1 Random oversampling

Like random undersampling, random oversampling is a simple yet effective approach to resampling. Here, one chooses members from the minority class at random; these randomly chosen members are then duplicated and added to the new training set. One must note two things when randomly oversampling. First, one chooses documents randomly from the original training set, not the new training set. To do otherwise would bias the randomness of selection. Second, one always randomly oversamples with replacement. If we were to randomly oversample without replacement, we would deplete all members of the minority class long before we reached the desired balance between the majority and minority class.

3.3.2 Neighbor Based Oversampling TEchnique (NBOTE)

This is an oversampling technique we are proposing in this thesis. Let us first describe the oversampling method, and then describe the intuition behind why we think this technique will do well.

First, for each minority class document in the training set, we find the class id of the ten most similar documents from the training set. Among these ten nearest neighbors, we find the number of documents belonging to the majority class. We now assign a probability that each document in the minority class training set will be oversampled. Let M be the number of minority class documents, d_{mi} represent the ith minority class document in the training set where i is a number between 1 and M, and nn-maj(d_{mi}) be the number of nearest neighbors to d_{mi} from the majority class. Then, the probability that the ith minority class document is selected to be oversampled is given by:

$$P(d_{mi}) = \frac{1 + nn - maj(d_{mi})}{M + \sum_{j=1}^{M} nn - maj(d_{mj})}$$

The intuition behind this oversampling scheme is simple. Minority class documents that are surrounded by majority class documents are more likely to be misclassified. So, the more majority class documents that are closest to a minority class document d_{mi}, the higher the probability that d_{mi} will be selected for oversampling. However, even if none of the ten nearest neighbors of a minority class document belong to the majority class, there is still a probability the document will be selected for oversampling. That is, even if nn-maj(d_{mi}) is zero, P(d_{mi}) is not. We expect higher recall but lower precision with this approach.

Finally, we note that there are some ways this scheme could possibly be improved. First, one needs to choose the number of neighbors to examine. We have arbitrarily chosen to always use ten. Second, one could weight the probabilities that a document is chosen differently. Third, we examine only the number of neighbors from the majority class, not the distance between the documents. This could potentially lead to serious overfitting. For example, let us examine two documents that have five neighbors from the minority class and five documents from the majority class. Clearly, if the five nearest neighbors of one document were from the minority class and the five nearest neighbors of the second document were from the majority class, the two documents should probably not be treated equally. (One encounters a similar dilemma in kNN when deciding whether to weight the votes of each neighbor by their distance.) Finally, one modification to the above scheme that we do not consider is to eliminate some of the randomness in selecting a document to oversample. For example, we could oversample each document in the minority class a set number of times based on the class membership of their nearest neighbors. The more nearest neighbors a document had that belonged to the majority class, the more times we would oversample the minority document. However, we ignore these concerns in the current study simply to examine the base potential of this method. If promising, future work could adapt some of these refinements.

3.3.3 Generative Oversampling

This is an oversampling technique we are proposing in this thesis. This oversampling method is very simple, but, to the best of our knowledge, has not been proposed in the academic community. To increase the number of minority class documents, we simply generate completely new documents based on the probability that a word will appear in a document from the minority class.

We first make the assumption that documents are distributed in a multinomial distribution. We also assume that the probability that one word appears in a document is independent of the appearance of other words in the document. Instead, the probability that a word will appear is dependent only on which class the document is from. While these assumptions may not be true, these are precisely the assumptions one makes when applying the multinomial naïve Bayes classification model, an algorithm that is very popular for text classification. In fact, our approach is inspired by the multinomial naïve Bayes classifier. Empirically, these assumptions have been shown to perform well despite their simplicity, so we continue based on these assumptions.

Given the documents from the minority class in our training set, we estimate the multinomial distribution of word appearances in a minority class document. We can express this formally as follows. The probability that a word w_j appears in a minority class document is given by:

$$P(w_j|c_{min}) = \frac{1 + a1}{|V| + a2}$$

where a1 is the number of times w_j appears in the minority class, a2 is the total number of words that appear in the minority class, |V| is the size of the vocabulary of the dataset, and c_{min} is the minority class.

We next find the average number of words that appear in documents from the minority class. Let us call this quantity N. We can now generate a document of length N by selecting words probabilistically N times from the multinomial distribution given above.

In some domains, it may be hard to find a probability distribution with which to model the training data. However, in any domain where naïve Bayes is used, the same distribution used for the naïve Bayes model can potentially be used to generate new minority class data. We leave empirical testing of this claim on lower dimensional data modeled with other distributions (e.g. Gaussian) to future work.

3.3.4 Perturbed Random Oversampling

This is an oversampling technique we are proposing in this thesis. This technique is a combination of random oversampling and generative oversampling. In particular, we want to slightly perturb documents chosen via random oversampling so that we do not place exact duplicates of existing minority class documents into the new training set. In contrast, in random oversampling, exact duplicates of the original training set are placed into the new training set when we use random oversampling. Depending on the classifier, this may lead to serious overfitting. For example, the classifier may end up remembering a record from the minority class simply because the classifier sees this same record several times. Thus, as stated in [Chawla02], exactly replicating a document may lead a classifier to assign false significance to a given record. On the other hand, replicating documents may add very little new information to the classification algorithm. For example, in the case of SVMs, the decision boundary between the majority and minority class depends only on those documents close to the decision boundary (i.e. the support vectors). Replicating a document that is not a support vector may change the prior probabilities of the document classes, but does nothing to change the decision boundary learned by an SVM.

A good example of a published oversampling technique which modifies existing documents before adding them to the training set is SMOTE [Chawla02]. Since SMOTE [Chawla02] creates new minority class examples based on existing minority class examples, this could be why SMOTE outperforms random oversampling.

Let us now describe the process of perturbed random oversampling. As in random oversampling, a document from the minority class is randomly selected from the training set. We also generate a completely new minority class document as described in generative oversampling. Our goal is to slightly perturb the existing document selected via random oversampling with values from the new, artificial minority class document created via generative oversampling. Let d_o be our original document, d_a be our artificial document, d_n be the new document we are creating, and $d(w_i)$ be the ith word from document d. Then, for each word w_i in d_a, there is a 60% chance that $d_n(w_i) = d_o(w_i)$, 10% chance that $d_n(w_i) = d_a(w_i)$, and 30% chance that $d_n(w_i) = (d_o(w_i) + d_a(w_i))/2$. These three probabilities are chosen so that the majority of word frequencies in the original document would remain unchanged, some of the word frequencies would be slightly changed, and a few of the word frequencies would be completely different. It is possible that we would get better results if these three probabilities were learned empirically. However, the goal is to slightly perturb existing documents, meaning that our new documents should be very similar to existing minority class documents. In comparison, when we use generative oversampling, we could create documents that are completely unlike any documents in the training set. We expect that generative oversampling will perform well when the minority class is distributed unimodally, while perturbed random oversampling should perform well when the minority class is distributed multimodally.

3.3.5 Perturbed NBOTE

This is an oversampling technique we are proposing in this thesis. This technique is similar to perturbed random oversampling, except for the fact that we perturb documents selected via NBOTE instead of random oversampling. For completeness sake, we will describe perturbed NBOTE formally. Let d_o be our original document selected via NBOTE, d_a be our artificial document created via generative oversampling, d_n be the new document we are creating, and $d(w_i)$ be the ith word from document d. Then, for each word w_i in d_a, there is a 60% chance that $d_n(w_i) = d_o(w_i)$, 10% chance that $d_n(w_i) = d_a(w_i)$, and 30% chance that $d_n(w_i) = (d_o(w_i) + d_a(w_i))/2$. The motivation for these probabilities is exactly the same as in perturbed random oversampling.

3.3.6 Synthetic Minority Oversampling TEchnique (SMOTE)

SMOTE was first introduced in [Chawla02]. The SMOTE algorithm is simple yet effective, outperforming random oversampling in several low dimensional problems. In particular, SMOTE is one of the few resampling techniques that has consistently outperformed random undersampling and random oversampling in terms of performance.

In the SMOTE algorithm, for each minority class document in the training set, find the 5 nearest neighbors from the remaining minority class documents in the training set. To create a new minority class document, we randomly select one of the five nearest neighbors. The new minority class document is a randomly selected point on the hyperplane joining the two minority class documents.

3.3.7 SMOTE2

This oversampling technique is a small modification of the SMOTE algorithm. Instead of selecting a point on the hyperplane between two minority class documents, we select a point on the hypercube between two minority class documents. The goal of using this oversampling technique is to see what happens when we create new minority class documents which are not as similar to the two original minority class documents. However, since we are in high dimensional space, we expect that SMOTE and SMOTE2 will perform approximately the same.

3.3.8 nonnSMOTE (Not Nearest Neighbor SMOTE)

This technique is another variation on the original SMOTE algorithm. However, we do not require that SMOTED documents be nearest neighbors. Instead, we randomly choose two minority class documents from the training set, and then randomly find a point on the hyperplane joining the two documents. The motivation behind this modification (along with nonnGAIOTE described in 3.3.10) is to examine the effect of combining minority class documents which may or may not be very similar to each other.

3.3.9 Genetic Algorithm Inspired Oversampling TEchnique (GAIOTE)

This is an oversampling technique inspired by genetic algorithms that we are proposing in this thesis. We first randomly select a minority class document. Let us call this document parent1. We now need to select a second document, parent2. We randomly select parent2 from the 5 nearest minority class neighbors of parent1. We now create a document called child based on parent1 and parent2. Specifically, given w_i, the ith word in the vocabulary, and $d(w_i)$, the

frequency that w_i occurs in document d, there is a 50% probability that child(w_i) = parent1(w_i) and a 50% probability that child(w_i) = parent2(w_i). We add the child of parent1 and parent2 to our new training set containing our oversampled documents.

3.3.10 nonnGAIOTE (Not Nearest Neighbor GAIOTE)

This technique is similar to GAIOTE described above except that parents do not have to be neighbors in order to produce children. Thus, we remove the complexity of finding the minority class neighbors to a given minority class document. The goal of using nonnSMOTE and nonnGAIOTE is to test how important it is for documents to be near one another when combining documents to form new documents. If we obtain similar results without having to find nearest neighbors, then we have saved computation time on the order $O(m^2)$, where m is the number of minority class documents. However, since m is typically rather small, this may or may not be a useful saving of time.

4.0 Experimental setup

Our experimental setup is as follows. For each dataset, we use 50% of the data as the training set and the remainder as the test set. Specifically, we randomly place 50% of each class of the original dataset into the training set. Thus, in both the training and test set, we maintain the same class prior probability as in the original dataset. We choose the class with the smallest prior probability as our target class, and conglomerate all other classes into the majority class. We perform this process ten times so that we have ten splits of training and test data. In particular, the same training/test splits are used for all resampling experiments in order to keep the results fair.

We apply each of our resampling methods to each train/test split. Specifically, we resample until 50% of the data in the training set consists of the minority class. After resampling, we classify using either multinomial naïve Bayes, kNN, or SVMs. While there are no parameters to adjust when using naïve Bayes, kNN and SVMs both require additional parameters. We use k=1, 5, 15, 25, and 35 for kNN. For SVMs, we use a linear kernel. The implementation of the SVM classifier is via SVMlight [Joachims99]. Finally, after classification, we average the classification results over all ten training/test splits.

We should note that we might get very different results if we change the level of resampling. In particular, some resampling methods might perform better with a higher or lower amount of resampling. It is also quite likely that resampling until the minority class data makes up 50% of the training data does not yield optimal results [Japkowicz04]. In particular, to reach this level of balance, we must either oversample the minority class many times or undersample many majority class documents. Those oversampling methods that duplicate members of the minority class will be more prone to overfitting, while all undersampling methods must eliminate a large amount of potentially useful information. Thus, one needs to keep in mind that the results below are for this given level of resampling.

4.1 Description of classification algorithms

Many past studies have focused on the effects of resampling on a single classification algorithm. In particular, C4.5 and kNN have been widely studied in the past. However, to determine if different resampling techniques yield better performance when coupled with different classifiers, we need to combine the various possible resampling techniques with various different classification algorithms. Thus, we study the effects of various resampling techniques on three classifiers: multinomial naïve Bayes, k nearest neighbor, and SVM. All three of these

classifiers are very popular and well studied in the realm of text classification. However, only kNN has received much attention in the realm of imbalanced datasets. While we are unaware of many studies using naïve Bayes in an imbalanced dataset scenario, we choose to include multinomial naïve Bayes because it is an extremely popular text classification algorithm that is extremely susceptible to class imbalance. [Japkowicz01] studied SVMs , but without conclusive results for how SVMs were affected by resampling.

A comprehensive discussion of multinomial naïve Bayes, kNN, and SVMs is beyond the scope of this thesis. Instead, we will give a very brief overview of each, paying particular attention to aspects of the algorithm that make it susceptible or immune to the imbalanced dataset problem. Those interested in more information should look at [McCallum98] and [Joachims98] for excellent discussions of the subject, particularly with regards to text classification.

4.1.1 Multinomial Naïve Bayes

The multinomial naïve Bayes classifier is an extremely popular yet simple algorithm used in text classification. Even though several other algorithms, such as kNN and SVM, often outperform multinomial naïve Bayes in terms of classification accuracy, its simplicity and speed make it very popular.

Unfortunately, the naïve Bayes classifier is unsuited to the imbalanced dataset problem because it takes into account the prior probabilities of classes in the training data when classifying new, unseen documents. Thus, in the not uncommon case where more than 99% of the data is made up of the majority class, the probability a document came from the minority class model must be more than 99 times greater than the probability that a document came from the majority class model in order for that document to be classified as the minority class. In a two class problem, this means the probability a document came from the majority class given the words in that document must be less than the prior probability of a document coming from the minority class.

One common solution is to shift the decision boundary between the majority and minority class by weighting the prior probabilities. As stated, however, we ignore cost sensitive classifiers for the sake of this study, leaving a comparison with cost sensitive approaches to future work.

4.1.2 k Nearest Neighbor

The k nearest neighbor classification algorithm is an instance-based learning method first proposed in [Cover67]. kNN is simple yet efficient and has been used in the past on text classification. In particular, kNN is popular since the only parameters one needs to choose are k and an appropriate distance metric.

We use a variant of kNN called weighted-distance based kNN. In this scheme, one first finds the k nearest neighbors in the training set to the document that one wishes to classify. Each of the k nearest neighbor then "votes" that the current document shares the same class. In the weighted-distanced based kNN scheme, the number of votes that a nearest neighbor gets to cast in the current decision is the same as the similarity between the nearest neighbor and the current document. Thus, we give the most weight to documents that are most similar to the current document.

kNN looks at local neighborhoods of the input space. Thus, those resampling methods that choose documents to resample based on local criteria should couple well with kNN. It is also possible that if most instances of the minority class are very close together, then the kNN classifier will do well even without resampling.

4.1.3 Support Vector Machines

SVMs are a very popular classification algorithm that was first used for text classification in [Joachims98]. Briefly, an SVM attempts to find a decision boundary between two classes by finding a hyperplane that best separates the two classes in some kernel space. This kernel space is one of the parameters that one needs to tune. In this experiment, we simply use a linear kernel, which has been shown to work well with text.

A thorough discussion of SVMs is beyond the scope of this thesis. Instead, we will discuss SVMs in light of the imbalanced dataset problem. First, the SVM classifier is naturally suited to two class problems. Since most researchers experimentally study imbalanced datasets in a two class setting by creating a single majority class and a single minority class, the two class nature of SVMs fits nicely. Second, the SVM classifier creates the optimal separating hyperplane based on only those documents on or near the boundary of the hyperplane. These documents are called support vectors. If a resampling method does not change the support vectors, the model built by the SVM will not change.

4.2 Datasets

One thing that we do not vary in this experiment is which classes to use as the minority class and which classes to use as the majority class. All of our datasets are split into more than two classes. We choose the smallest class as the minority class, and conglomerate all remaining classes into the majority class. This approach has been taken in the past by several others. One interesting approach that would perhaps be more statistically accurate would be to let each defined class be the minority class once, while conglomerating all remaining classes as the majority class. However, since we are averaging the results of ten random splits of the data into training and test documents, we consider our results to be statistically accurate enough.

We use eight publicly available text datasets which have been used by others in several previous studies. In particular, we use the version of the datasets used in [Zhong03]. That is, we use the same preprocessing steps as [Zhong03]. We use the standard vector space model representation for text and tf-idf weighting. Finally, to eliminate the effects of document length, we normalize each document vector using the L2-norm. However, we normalize and take tf-idf weighting after resampling has finished.

An interesting note is that many of the resampling methods produce document vectors with non-integer word frequencies. After normalizing, however, all word frequencies take values between zero and one, so we do not consider resampling methods that return non-integer word frequencies in the realm of text classification to be invalid. In addition, none of the resampling methods produce negative word frequencies, so we do not have any document vectors which represent documents which cannot possibly occur.

The details of the datasets are given in Table 1.

Dataset	Num Docs	Vocab	Minority class size	Fraction minority class
classic	7094	41681	1033	0.146
hitech	2301	10080	116	0.050
k1b	2340	21839	60	0.026
la1	3204	31472	273	0.085
la2	3075	31472	248	0.081
ohscal	11162	11465	709	0.064
reviews	4069	18483	137	0.034
sports	8580	14870	122	0.014

Table 1: The number of documents, vocabulary size, number of minority class documents, and the fraction of documents making up the minority class in each of our eight datasets.

In our experiment, we choose our feature vector based on traditional feature selection methods. In [Zheng04], the authors show that, in the case of imbalanced text classification, it is better to take a combination of words that are highly indicative and words that are not indicative of class membership. In [Serrano04], the authors use feature selection guided by genetic algorithms to improve text classification using naïve Bayes. However, we still use traditional feature selection methods in order to better isolate the effects of resampling on classification of high dimensional data. An interesting area of future work would be to examine the effect of [Zheng04]'s and [Serrano04]'s methods of feature selection on resampling, and whether one can improve the results presented both here and in [Zheng04] and in [Serrano04].

4.3 Evaluation metric

As is customary in other studies on imbalanced datasets, we start this section by pointing out that classification accuracy is not an appropriate metric to evaluating classification performance in the imbalanced dataset scenario. Let the overall classification accuracy be measured as the percent of the documents in the test set which were identified correctly by the classifier. Unfortunately, this is not a good metric for the imbalanced dataset scenario. Let us assume that only 1% of the data in the test set is from our target class. If the classifier always guesses that a document is from the majority/non-target class, then we automatically achieve 99% classification accuracy. This does not reflect the fact that we have misclassified every single instance of the minority class. In particular, this does not reflect the fact that it is usually more important to classify the minority class correctly.

We choose to use the measures of precision, recall and f-measure. These measures have been used in several studies in the past on both imbalanced datasets and text. Most importantly, many studies point out that these measures are not biased because of the class imbalance problem. However, precision and recall need to be considered simultaneously for this to be true.

Let a document from the minority class which is correctly classified be a true positive (TP), and a document from the minority class which is incorrectly classified be a false negative (FN). Let a document from the majority class which is correctly classified be a true negative (TN), and a document from the majority class which is incorrectly classified be a false positive (FP). Then, precision and recall are defined as follows:

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

$$Recall = \frac{TP}{TP + FN}$$

$$F - measure = \frac{2 * precision * recall}{precision + recall}$$

Precision, recall, and f-measure all take values ranging from zero to one, where one is the best a classifier can do on that particular metric. Intuitively, precision tells us what fraction of the documents classified as the target class are actually from the target class, while recall tells us what fraction of the target class was correctly identified. In general, there is a tradeoff between the precision and recall that can be achieved. That is, one must typically sacrifice high precision for high recall and vice versa. Thus, the f-measure is a convenient way of looking at the tradeoff between precision and recall in one single value. In a sense, the f-measure measures the balance between precision and recall.

5.0 Results

For each individual classifier, we will first describe general trends in our results and then analyze these general trends. In particular, we will address the issues of whether certain resampling techniques tend to perform well when coupled with certain classifiers, whether oversampling performs better than undersampling for a given classifier, whether oversampling techniques which create new data points (e.g. SMOTE, generative oversampling) outperform oversampling techniques which replicate data points (e.g. random oversampling, NBOTE), and whether oversampling techniques which combine minority class documents which are nearest neighbors (e.g. SMOTE, GAIOTE) outperform their counterparts which do not require documents to be nearest neighbors (e.g. nonnSMOTE, nonnGAIOTE). Finally, we will compare the best f-measure returned by each classifier on each dataset before and after resampling to compare the increase of classifier performance due to resampling.

For individual results on each dataset, please see Appendix A. In particular, Appendix A contains all of the numerical values for precision, recall, and f-measure in tabular format, along with a description of specific trends for each classifier on each dataset. Here, we merely state whether certain resampling techniques tend to produce high precision or high recall when compared to other resampling techniques without mention of the actual numerical value.

To make our results easier to digest, we will also provide summary statistics for each classifier. Assuming that it is equally important to have high precision and high recall, we evaluate classifiers based on f-measure. For each dataset, there are eighteen possible resampling techniques we can use. We rank the resulting f-measure for each resampling technique, with a rank of 1 corresponding to the highest f-measure for that dataset and a rank of 18 for the lowest f-measure. We derive two summary statistics from these rankings. The first is the average ranking that a resampling technique receives on all eight datasets. The second statistic is simply a count of how many times a resampling technique produced an f-measure that was ranked in the top three, ranked four through nine, ranked ten through fifteen, and ranked sixteen through eighteen. Let us call these four groupings tier I, tier II, tier III, and tier IV. In the tables in this section, we simply list a count of how many datasets a resampling technique yielded f-measure in tier I (rank 1-3), tier II (rank 4-9), tier III (rank 10-15), and tier IV (rank 16-18). The average ranking and tier system will be used as a convenient way of accessing roughly how well certain resampling techniques performed when combined with certain classifiers.

5.1 Multinomial Naïve Bayes

Let us first examine the performance of multinomial naïve Bayes without the use of resampling. Without resampling, naïve Bayes performs very poorly on our datasets and rarely identifies a document as the minority class. In fact, for some of the datasets, naïve Bayes always classifies documents in the test set as belonging to the majority class. That is, there are no true positives or false positives identified. With resampling, however, the performance of naïve Bayes increases dramatically. Not only are documents identified as the minority class, but naïve Bayes becomes competitive with kNN and SVMs as we shall see in section 5.4.

In general, the performance of resampling techniques coupled with naïve Bayes seems dataset dependent. However, there are still some apparent trends. Generative oversampling always yields high precision at the cost of recall. Since generative oversampling creates new documents based on the multinomial distribution, it is possible that generative oversampling helps multinomial naïve Bayes to learn a more accurate model of the minority class, thus increasing precision. However, since we are creating new documents based only on the documents in the training set, we are probably overfitting the training set, thus resulting in lower recall.

Undersampling tends to outperform oversampling in terms of recall, but at a very high cost of precision. Random oversampling performs fairly, yielding neither the best or worst f-measure when compared to other oversampling techniques. Oversampling techniques yield much better f-measure when compared to undersampling. Finally, none of the undersampling techniques significantly outperform random undersampling. That is to say, undersampling seems to be a very poor choice when using naïve Bayes.

There is no trend as to whether techniques which create new minority class documents based on nearest neighbors (e.g. SMOTE, GAIOTE) outperform their counterparts which do not require documents to be neighbors (e.g. nonnSMOTE, NONNGAIOTE). However, there is a trend when comparing oversampling techniques which create new minority class documents with oversampling techniques which merely replicate existing minority class documents. In general, techniques which create new minority class documents outperform techniques which simply replicate. In particular, perturbed random oversampling beats random oversampling and perturbed NBOTE beats NBOTE. Strangely, SMOTE and its variants only perform well on some datasets, often being beaten by random oversampling.

Summary statistics are provided in Table 2. Based on these statistics, we see that the best resampling techniques are perturbed NBOTE and perturbed random oversampling. That is, our

best performing resampling techniques are the two techniques that perturb existing minority class documents in order to create new data for the training set. Generative oversampling, which creates completely new minority class documents, also does quite well, but there are several datasets on which it performs poorly. As stated in section 3, generative oversampling will most likely do poorly if the minority class is multimodal. Finally, techniques which combine two existing documents also perform well, but not as well as generative oversampling. In particular, GAIOTE, SMOTE, and SMOTE2 all perform well. Their counterparts which do not take nearest neighbors into consideration perform poorly comparatively. Most likely, these techniques perform well when combined with the multinomial naïve Bayes classifier since we learn a more general model for the minority class after introducing new documents.

Naïve Bayes: Summary Statistics					
	Average rank	**Tier I**	**Tier II**	**Tier III**	**Tier IV**
no_resampling	17.75	0	0	0	8
GAIOTE	4.5	3	4	1	0
nonnGAIOTE	7.5	1	6	1	0
nonnSMOTE	7.25	0	7	1	0
NBOTE	6.625	0	7	1	0
generative	5.875	4	1	3	0
pertNBOTE	2.875	6	2	0	0
pertrand	3.75	5	2	1	0
randover	6	0	8	0	0
SMOTE	5.125	3	5	0	0
SMOTE2	5.625	2	6	0	0
Distant1	15.25	0	0	4	4
Distant2	14.75	0	0	8	0
Distant3	16.75	0	0	0	8
NearMiss1	14.375	0	0	5	3
NearMiss2	12.25	0	0	7	1
NearMiss3	12.625	0	0	8	0
randunder	12.125	0	0	8	0

Table 2: Summary statistics for the multinomial naïve Bayes classifier

5.2 kNN

When using kNN, one must often choose k empirically. That is, we choose k by observing which choice of k yields the best classification. In this case, let us use f-measure to gauge classification accuracy. When examining the best choice of k in the case where we do not resample, we often find that this k is not the best choice of k in the case where we do resample. That is, resampling may change which k is optimal for a given dataset. Thus, in our results, we will simply look at the highest f-measure obtained for a given resampling method and dataset without noting which value of k was used.

For the kNN classifier, we see some interesting general trends. In this case, simple approaches seem to work best. In particular, kNN works quite well on several text datasets without resampling. Since kNN classifies documents based on local neighborhoods of the data, the minority class consisted of documents which were very similar to each other. When resampling does help, variants of SMOTE or random oversampling seem to work best. Sophisticated undersampling techniques do not seem to work well at all. It is a bit surprising that the Distant undersampling techniques do not outperform random undersampling since the Distant techniques attempt to eliminate majority class documents that are very close to the minority class.

Based on summary statistics, SMOTE outperforms nonnSMOTE and GAIOTE outperforms nonnGAIOTE. This makes sense since the documents created by SMOTE and GAIOTE will be very similar and possible neighbors of both of the two parent documents which were used to create the new document. Random oversampling tends to outperform perturbed random oversampling while NBOTE tends to outperform perturbed NBOTE. Given that the majority of the top performing resampling methods (i.e. those with high average rank and high marks in tier I) are techniques which either keep the minority class the same (no resampling), add duplicates of the minority class (random oversampling, NBOTE) or create documents which are very similar to existing minority class documents (variants of SMOTE), it seems that the best oversampling techniques to use with kNN are techniques which add new minority documents as close to existing minority class documents as possible. Techniques such as generative oversampling and perturbed random oversampling carry much less of a guarantee as to how similar new minority class documents will be to existing minority class documents.

knn: Summary Statistics					
	Average rank	Tier I	Tier II	Tier III	Tier IV
no_resampling	4.125	5	2	1	0
GAIOTE	7.75	0	7	1	0
nonnGAIOTE	9.75	0	4	4	0
nonnSMOTE	3.875	3	5	0	0
NBOTE	4.875	3	5	0	0
generative	5.875	3	4	1	0
pertNBOTE	8.5	0	4	4	0
pertrand	7.625	1	4	3	0
randover	4.125	2	6	0	0
SMOTE	3.125	5	3	0	0
SMOTE2	6.875	2	4	2	0
Distant1	16.25	0	0	2	6
Distant2	15.625	0	0	3	5
Distant3	17.75	0	0	0	8
NearMiss1	14.125	0	0	7	1
NearMiss2	13.5	0	0	6	2
NearMiss3	14.25	0	0	6	2
randunder	13	0	0	8	0

Table 3: Summary statistics for the kNN classifier; for each combination of resampling method and dataset, we take the value of k that yields the highest f-measure as the best value of k (i.e. we examine only the best f-measure for each combination of resampling method and dataset)

5.3 SVM

In general, the case where we do not resample tends to produce the highest precision. SMOTE and its variants also produce high precision, but with poor recall. In fact, SMOTE and its variants rarely produce precision higher than the case where we do not resample, thus resulting in poor f-measure. However, because the case where we do not resample yields such high precision, few techniques consistently beat no resampling with regards to f-measure. In particular, random undersampling and NearMiss3 are the only consistently viable undersampling techniques. We suspect that NearMiss3 works well when coupled with SVMs since NearMiss3 strives to keep documents that are near the decision boundary between minority and majority class documents. In particular, the goal of NearMiss3 is to ensure that all minority class documents are surrounded by majority class documents. Out of our three classifiers, SVMs are the only classifier where undersampling techniques are promising.

Looking at GAIOTE vs nonnGAIOTE and SMOTE vs nonnSMOTE, there is again no dataset specific trend. In particular, based on our summary statistics in table 4, SMOTE tends to outperform nonnSMOTE, but nonnGAIOTE tends to outperform GAIOTE. It is clear, however, that techniques which merely replicate members of the minority class do not perform as well as techniques which create new minority class documents. In particular, generative oversampling, perturbed random oversampling, and perturbed NBOTE, and nonnGAIOTE perform well based on our summary statistics.

SVM: Summary Statistics					
	Average rank	Tier I	Tier II	Tier III	Tier IV
no_resampling	9.625	0	4	4	0
GAIOTE	6.125	1	7	0	0
nonnGAIOTE	4.375	3	5	0	0
nonnSMOTE	11	0	1	7	0
NBOTE	7.5	1	5	2	0
generative	2.375	5	3	0	0
pertNBOTE	3.75	4	4	0	0
pertrand	3.5	5	3	0	0
randover	8	1	4	3	0
SMOTE	10.25	0	4	4	0
SMOTE2	12.125	0	0	8	0
Distant1	15.125	0	0	5	3
Distant2	14.375	0	0	6	2
Distant3	17.625	0	0	0	8
NearMiss1	16	0	0	3	5
NearMiss2	10.625	0	3	4	1
NearMiss3	11.875	2	1	0	5
randunder	6.75	2	4	2	0

Table 4: Summary statistics for the SVM classifier

5.4 Comparison of classifier performance

Let us conclude this section by comparing the best f-measure returned by each classifier on each dataset. Table 5 contains the highest f-measure for each classifier before resampling. Table 6 contains the highest f-measure for each classifier after resampling. The classifiers on each dataset are listed in order of highest to lowest f-measure.

On every single dataset in our experiment, we are able to obtain a higher f-measure after resampling. Moreover, the best choice of classifier before resampling is not necessarily the same as the best choice of classifier after resampling. Naïve Bayes clearly sees the greatest improvement in performance after resampling. Before resampling, the naïve Bayes method rarely classifies a document as the majority class. After resampling, however, naïve Bayes becomes competitive with both kNN and SVM. The performance of SVMs with regards to f-measure also improves quite well after resampling. However, kNN sees no improvement after resampling on half of the datasets. Since kNN classifies documents based on local criteria, this could be due to the fact that minority class documents in these datasets are all very similar to each other.

It is apparent from table 6 that the best oversampling techniques tend to be those that create new minority class documents instead of merely replicating existing minority class documents. In particular, generative oversampling, perturbed NBOTE, variants of SMOTE, and perturbed random oversampling appear quite frequently in table 6. kNN coupled with no resampling is the best kNN approach on four datasets, but is always outperformed by SVMs with some type of resampling. Finally, on the reviews dataset, the best combination of classifier and resampling technique is SVM with random undersampling.

Based on these results, it is possible to significantly improve performance of multinomial naïve Bayes, kNN, and SVMs on text datasets by using resampling techniques.

Best results before resampling				
Dataset	**Classifier**	**F-measure**	**Precision**	**Recall**
classic	svm	0.963 ± 0.006	0.999 ± 0.001	0.930 ± 0.010
	knn(k=1)	0.929 ± 0.009	0.991 ± 0.003	0.874 ± 0.015
	nb	0.565 ± 0.010	1.000 ± 0.000	0.394 ± 0.009
hitech	knn(k=1)	0.503 ± 0.042	0.554 ± 0.046	0.464 ± 0.053
	svm	0.374 ± 0.064	0.896 ± 0.075	0.240 ± 0.051
	nb	NaN	NaN	0.000 ± 0.000
k1b	knn(k=1)	0.546 ± 0.076	0.594 ± 0.108	0.520 ± 0.098
	svm	0.230 ± 0.087	0.663 ± 0.274	0.143 ± 0.059
	nb	NaN	NaN	0.000 ± 0.000
la1	knn(k=1)	0.456 ± 0.045	0.469 ± 0.045	0.445 ± 0.051
	svm	0.326 ± 0.035	0.811 ± 0.051	0.204 ± 0.026
	nb	NaN	NaN	0.000 ± 0.000
la2	knn(k=5)	0.565 ± 0.024	0.727 ± 0.056	0.465 ± 0.029
	svm	0.491 ± 0.043	0.932 ± 0.034	0.335 ± 0.039
	nb	NaN	NaN	0.000 ± 0.000
ohscal	svm	0.718 ± 0.021	0.858 ± 0.018	0.618 ± 0.030
	knn(k=5)	0.600 ± 0.022	0.768 ± 0.027	0.492 ± 0.027
	nb	0.026 ± 0.006	1.000 ± 0.000	0.013 ± 0.003
reviews	knn(k=5)	0.876 ± 0.028	0.970 ± 0.028	0.800 ± 0.042
	svm	0.847 ± 0.032	1.000 ± 0.000	0.735 ± 0.048
	nb	NaN	NaN	0.000 ± 0.000
sports	svm	0.898 ± 0.034	0.979 ± 0.016	0.831 ± 0.057
	knn(k=5)	0.895 ± 0.037	0.942 ± 0.037	0.856 ± 0.066
	nb	NaN	NaN	0.000 ± 0.000

Table 5: Results before resampling listed in order of highest to lowest f-measure; for kNN, we list only the best result, regardless of k; on several datasets, the naïve Bayes classifier was unable to identify any documents as the minority class (zeros true positives and zero false positives) which we have indicated by the entry "NaN" for f-measure and precision

Best results after resampling					
Dataset	**Classifier**	**Resampling Method**	**F-measure**	**Precision**	**Recall**
classic	svm	generative	0.985 ± 0.002	0.988 ± 0.004	0.982 ± 0.005
	nb	pertNBOTE	0.978 ± 0.004	0.969 ± 0.008	0.986 ± 0.004
	knn(k=35)	nonnSMOTE	0.967 ± 0.005	0.968 ± 0.007	0.966 ± 0.008
hitech	nb	generative	0.529 ± 0.029	0.707 ± 0.075	0.428 ± 0.046
	knn(k=1)	SMOTE	0.505 ± 0.059	0.512 ± 0.064	0.502 ± 0.068
	svm	nonnGAIOTE	0.501 ± 0.078	0.718 ± 0.084	0.390 ± 0.085
k1b	svm	generative	0.687 ± 0.063	0.767 ± 0.084	0.627 ± 0.072
	nb	Pertrand	0.608 ± 0.038	0.439 ± 0.041	0.993 ± 0.014
	knn(k=5)	Pertrand	0.605 ± 0.053	0.499 ± 0.048	0.770 ± 0.071
la1	svm	pertNBOTE	0.561 ± 0.030	0.818 ± 0.040	0.428 ± 0.036
	nb	pertNBOTE	0.551 ± 0.035	0.435 ± 0.044	0.758 ± 0.028
	knn(k=1)	SMOTE	0.468 ± 0.043	0.438 ± 0.031	0.506 ± 0.062
la2	svm	pertNBOTE	0.653 ± 0.032	0.848 ± 0.035	0.531 ± 0.035
	knn(k=5)	no_resampling	0.565 ± 0.024	0.727 ± 0.056	0.465 ± 0.029
	nb	pertNBOTE	0.559 ± 0.030	0.436 ± 0.039	0.786 ± 0.027
ohscal	svm	generative	0.744 ± 0.010	0.758 ± 0.022	0.732 ± 0.029
	knn(k=5)	no_resampling	0.600 ± 0.022	0.768 ± 0.027	0.492 ± 0.027
	nb	generative	0.588 ± 0.011	0.452 ± 0.015	0.842 ± 0.018
reviews	svm	randunder	0.940 ± 0.016	0.963 ± 0.019	0.918 ± 0.025
	knn(k=5)	no_resampling	0.876 ± 0.028	0.970 ± 0.028	0.800 ± 0.042
	nb	pertNBOTE	0.793 ± 0.023	0.766 ± 0.037	0.825 ± 0.047
sports	svm	generative	0.944 ± 0.018	0.982 ± 0.008	0.908 ± 0.031
	nb	SMOTE2	0.914 ± 0.020	0.908 ± 0.035	0.921 ± 0.041
	knn(k=5)	no_resampling	0.895 ± 0.037	0.942 ± 0.037	0.856 ± 0.066

Table 6: Best results after resampling listed in order of highest to lowest f-measure

6.0 Future Work and Conclusions

6.1 Future Work

With regards to future work, there are several issues that need to be addressed empirically. The first issue is to what degree we need to resample the training set. In our experiment, we resample until there the majority and minority classes have equal prior probability. However, this may not yield optimal results when coupled with certain resampling techniques [Weiss03]. Thus, we need to examine empirical results at different levels of resampling. In particular, this could be a reason why undersampling techniques fare so poorly in our empirical experiments. Interestingly enough, in [Batista04], the authors also resample using undersampling and oversampling techniques until the majority and minority classes have equal class prior probabilities. Using C4.5 as a classifier, the authors find that oversampling techniques outperform undersampling techniques. This is contrary to several other studies such as [Drummond03] which find that undersampling tends to outperform oversampling when using C4.5. As stated, part of this could be due to different datasets. However, this only underlies the need to perform more experiments with different levels of resampling. Regardless, we still feel it is significant that random undersampling often outperforms undersampling techniques. This is consistent with past studies that have shown it is very difficult to outperform random undersampling with more sophisticated undersampling techniques.

Second, similar benchmarks need to be done on low dimensional data. In particular, we have introduced several oversampling techniques which have performed well on text. We need to see if these same techniques yield similar performance in low dimensional domains as well. We also need to examine techniques such as generative oversampling when data is distributed in different probability distributions.

Finally, resampling techniques need to be compared to other methods of dealing with imbalanced datasets such as cost sensitive learning. For example, [Chawla02]'s empirical study compared SMOTE to naïve Bayes with cost adjusted decision boundaries. While SMOTE outperformed cost adjusted naïve Bayes, this was in a low dimensional setting. We need to perform similar studies in a high dimensional setting as well.

6.2 Conclusion

In this study, we have attempted to benchmark several resampling techniques in the realm of text classification. In particular, we have compared results for several datasets with different attributes and several classification algorithms which create classification algorithms in very

different manners. For our experimental settings, we have seen that choosing an appropriate resampling technique and appropriate classifier is dataset dependent. This is not surprising given that in many domains, including text classification, there is no "best" classifier. However, we have seen that certain resampling methods seem to complement certain classifiers well.

Unfortunately, this means that choosing an appropriate resampling technique becomes one extra step in preprocessing that must be done with little prior knowledge. Future studies need to address how to choose a suitable resampling technique based on the characteristics of the target dataset. In conclusion, however, it is clear that resampling can significantly improve the performance of multinomial naïve Bayes, k nearest neighbors, and SVMs in the realm of text classification. Since multinomial naïve Bayes, kNN, and SVMs are all popular text classification algorithms, resampling should be considered as a preprocessing step when classifying text in an imbalanced dataset scenario.

Appendix: Additional Results

Sections A.1, A.2, and A.3 describe dataset specific results for multinomial naïve Bayes, kNN, and SVMs. Tables 7-13 list dataset specific results in numerical format.

A.1 Multinomial Naïve Bayes

Results for precision, recall, and f-measure are contained in Table 7.

On the classic dataset, naïve Bayes is able to identify documents from the minority class without resampling. In fact, we have the highest precision when we do not resample. However, without resampling, we obtain very poor recall. On the classic dataset, oversampling and undersampling techniques all yield high recall levels. In the case of precision, oversampling beats undersampling. In particular, several of the undersampling techniques yield very poor precision, thus resulting in poor f-measure as well.

On the hitech dataset, oversampling again beats undersampling in terms of precision. In particular, generative oversampling yields the highest precision by far. However, compared to the other oversampling techniques, generative oversampling has high precision at the cost of recall. None of the undersampling techniques perform very well, although one should note that random undersampling, Distant1, Distant2, and Distant3 have very high recall at the cost of very poor precision. In terms of f-measure, oversampling techniques clearly beat undersampling techniques.

On the k1b dataset, we again see that generative oversampling coupled with naïve Bayes yields extremely high precision. However, generative oversampling also yields the lowest recall. On this dataset, all of the resampling techniques except generative oversampling, NearMiss1, and NearMiss3 yield very high recall. Again, oversampling beats undersampling in terms of f-measure. In particular, perturbed random oversampling and perturbed NBOTE perform very well with regards to f-measure when compared to all other techniques.

On the la1 dataset, we see that generative oversampling yields the highest precision but the lowest recall. We also see that undersampling techniques yield higher recall than oversampling techniques, but lower precision as well. In particular, random undersampling beats all other undersampling techniques with regard to f-measure. Perturbed NBOTE and perturbed random oversampling outperform all other techniques by a significant margin with regards to f-measure.

On the la2 dataset, we again see similar trends. All oversampling techniques beat all undersampling techniques with regards to f-measure. Generative oversampling still yields the

highest precision at the cost of poor recall. Perturbed NBOTE and perturbed random oversampling still perform very well with regards to f-measure.

On the ohscal dataset, we see slightly different trends. Naïve Bayes with no resampling yields the highest precision, but at the cost of extremely poor recall. Among the resampling techniques, generative oversampling yields the highest precision at the cost of the lowest recall. However, the difference between the precision and recall of generative oversampling and the other resampling techniques is much less pronounced on the ohscal dataset. Once again, oversampling tends to beat undersampling with regards to precision and f-measure, while undersampling techniques tend to yield higher recall (except for NearMiss3). Finally, this is the first dataset where we see SMOTE performing well, although generative oversampling outperforms SMOTE with regards to precision and f-measure.

In the reviews dataset, we once again see familiar trends. Oversampling beat undersampling with regards to precision and f-measure, but loses to undersampling in terms of recall. Perturbed NBOTE and perturbed random oversampling both yield significantly higher f-measure than all other resampling techniques. Finally, generative oversampling once again yields the highest precision at the cost of low recall.

In the sports dataset, we see that SMOTE and its variants perform best in terms of precision and f-measure. Generative oversampling does not yield the highest precision, but still yields the lowest recall. Generative oversampling and perturbed NBOTE still perform well, but are significantly outperformed by all other resampling techniques. In particular, random oversampling outperforms perturbed random oversampling and NBOTE outperforms pertNBOTE in terms of precision and f-measure. However, oversampling techniques still beat undersampling techniques in terms of precision and f-measure.

A.2 k Nearest Neighbor

Results for kNN are in tables 8 through 12.

On the classic dataset, the best choice of k when we do not resample is 15. However, we obtain the best f-measure on the classic dataset when we combine oversampling technique with large k (that is, k equal to 35). In contrast, when we do not resample, setting k equal to 35 yields the fourth best f-measure. After resampling, nonnSMOTE, SMOTE, and SMOTE2 are the only techniques that beat out random oversampling, all with k equal to 35.

On the hitech dataset, the best choice of k without resampling is to set k equal to 1. With regards to f-measure, SMOTE with k equal to 1 is the only resampling technique that beats out

kNN with k equal to 1 and no resampling. However, this gain in f-measure is not much. Our third and fourth best results are obtained for nonnSMOTE with k equal 1 and random oversampling with k equal to 1.

On the k1b dataset, setting k equal to 1 yields the best f-measure without resampling. However, the use of resampling results in several techniques which outperform this f-measure. In particular, setting k equal to 5 coupled with perturbed random oversampling, random oversampling, and NBOTE yield the top three f-measures.

On the la1 dataset, the top ten best performers are all for k equal to 1. In particular, the eighth best performer is for the case where we do not resample. The top three performers are SMOTE2, SMOTE, and random oversampling.

On the la2, ohscal, reviews, and sports datasets, the top performers are all for the case where we do not resample. In particular, resampling seems to hurt f-measure significantly. For example, on the ohscal dataset, we see a similar trend. The top three performers with regards to f-measure are all for the case where we do not resample. However, there is a significant drop in f-measure after the top three performers. In particular, we see that the best choice of k changes depending on whether or not one resamples.

A.3 SVMs

SVM results are listed in table 13.

In the classic dataset, all techniques (including no resampling) perform comparably except for Distant1, Distant2, NearMiss2, and Distant3. We get the best overall performance as measured by f-measure with generative oversampling, although random undersampling is not far behind.

On the hitech dataset, only nonnGAIOTE beats out random oversampling with regards to f-measure. Moreover, none of the undersampling techniques outperform either random undersampling or the case where we do no resampling. The undersampling techniques produce high recall at the cost of very low precision. Interestingly, we get the best precision when we do not resample. However, all of the oversampling techniques are able to yield a better f-measure then the case where we do not resample and when we use random undersampling.

On the k1b dataset, generative oversampling outperforms all other resampling techniques significantly when we look at precision and f-measure. We get the highest recall with undersampling techniques, but varying performance with regards to precision. In particular, the only undersampling technique to outperform random undersampling with regards to f-measure is

NearMiss2. In fact, only three oversampling techniques are able to outperform random undersampling, generative oversampling, perturbed NBOTE, and perturbed random oversampling. In this dataset, the appropriate resampling technique is able to significantly improve performance with regards to f-measure, although only generative oversampling outperforms the precision of the case where we have no resampling. This is mainly due to the fact that we get extremely poor recall when we do not resample.

On the la1 dataset, we see higher precision for oversampling techniques and higher recall for undersampling techniques. In terms of f-measure, however, oversampling techniques tend to beat undersampling techniques, even though random undersampling outperforms random oversampling. However, none of the undersampling techniques perform better with regards to f-measure than random undersampling. The top three techniques with regards to f-measure are perturbed NBOTE, perturbed random oversampling, and nonnGAIOTE.

On the la2 dataset, we again see that oversampling techniques tend to produce higher precision and undersampling techniques tend to produce higher recall. On this dataset, the highest precision is obtained by not resampling. SMOTE and its variants also produce fairly high precision compared to the case where we do not resample. However, several other techniques, including random oversampling and random undersampling, are able to produce higher f-measure. In particular, perturbed NBOTE, nonnGAIOTE, and perturbed random oversampling perform best with regards to f-measure. Finally, none of the undersampling techniques except random undersampling can produce a higher f-measure than the case where we do not resample.

In the ohscal dataset, we again see similar trends. That is, oversampling tends to produce higher precision then undersampling, while undersampling tends to produce higher recall. Oversampling techniques tend to fair better on the tradeoff between precision and recall, producing higher f-measure. In particular, generative oversampling and GAIOTE produce the highest f-measure. The case where we do not resample produces the highest precision, while SMOTE and its variants produce the next highest precision.

On the reviews dataset, we see several of the undersampling techniques performing quite well. Namely, random undersampling, NearMiss3, and NearMiss2 are the first, third, and fourth best performing techniques when looking at f-measure. The only other technique that outperforms no resampling with regards to f-measure is generative oversampling. Once again, undersampling tends to produce higher recall, while oversampling tends to produce higher precision. In particular, SMOTE and its variants and no resampling produce the highest precision.

On the sports dataset, generative oversampling and NearMiss3 perform best with regards to f-measure. Again, the case where we do not resample performs quite well with regards to f-measure, although most oversampling techniques are able to outperform no resampling with regards to precision. Once again, we see that oversampling beats undersampling with regards to precision, but typically produces lower recall.

Naïve Bayes: Precision								
	classic	**hitech**	**k1b**	**la1**	**la2**	**ohscal**	**reviews**	**sports**
no_resampling	1.000 ± 0.000	NaN	NaN	NaN	NaN	1.000 ± 0.000	NaN	NaN
GAIOTE	0.972 ± 0.007	0.286 ± 0.024	0.375 ± 0.026	0.351 ± 0.028	0.365 ± 0.024	0.414 ± 0.013	0.660 ± 0.042	0.865 ± 0.056
nonnGAIOTE	0.972 ± 0.009	0.298 ± 0.026	0.352 ± 0.024	0.309 ± 0.018	0.324 ± 0.022	0.391 ± 0.011	0.609 ± 0.041	0.795 ± 0.070
nonnSMOTE	0.966 ± 0.008	0.303 ± 0.030	0.375 ± 0.026	0.272 ± 0.015	0.289 ± 0.021	0.398 ± 0.012	0.637 ± 0.041	0.866 ± 0.039
NBOTE	0.963 ± 0.009	0.294 ± 0.025	0.360 ± 0.025	0.340 ± 0.026	0.347 ± 0.027	0.390 ± 0.012	0.654 ± 0.035	0.807 ± 0.061
generative	0.983 ± 0.006	0.707 ± 0.075	0.861 ± 0.102	0.830 ± 0.049	0.847 ± 0.052	0.452 ± 0.015	0.973 ± 0.045	0.863 ± 0.051
pertNBOTE	0.969 ± 0.008	0.351 ± 0.036	0.437 ± 0.047	0.435 ± 0.044	0.436 ± 0.039	0.396 ± 0.012	0.766 ± 0.037	0.731 ± 0.069
pertrand	0.974 ± 0.007	0.353 ± 0.039	0.439 ± 0.041	0.426 ± 0.031	0.422 ± 0.032	0.398 ± 0.011	0.748 ± 0.023	0.734 ± 0.073
randover	0.970 ± 0.008	0.304 ± 0.027	0.362 ± 0.027	0.327 ± 0.023	0.337 ± 0.023	0.393 ± 0.011	0.623 ± 0.037	0.807 ± 0.056
SMOTE	0.967 ± 0.007	0.295 ± 0.026	0.393 ± 0.025	0.311 ± 0.021	0.327 ± 0.020	0.424 ± 0.013	0.685 ± 0.056	0.910 ± 0.034
SMOTE2	0.967 ± 0.009	0.291 ± 0.025	0.392 ± 0.027	0.306 ± 0.021	0.326 ± 0.020	0.422 ± 0.014	0.678 ± 0.057	0.908 ± 0.035
Distant1	0.371 ± 0.029	0.079 ± 0.005	0.107 ± 0.009	0.156 ± 0.011	0.145 ± 0.009	0.130 ± 0.004	0.045 ± 0.000	0.024 ± 0.002
Distant2	0.688 ± 0.017	0.080 ± 0.003	0.101 ± 0.011	0.148 ± 0.011	0.132 ± 0.006	0.167 ± 0.005	0.066 ± 0.003	0.026 ± 0.003
Distant3	0.213 ± 0.022	0.054 ± 0.001	0.061 ± 0.007	0.131 ± 0.011	0.123 ± 0.008	0.096 ± 0.007	0.048 ± 0.003	0.017 ± 0.001
NearMiss1	0.889 ± 0.017	0.065 ± 0.014	0.326 ± 0.124	0.129 ± 0.009	0.122 ± 0.006	0.172 ± 0.004	0.126 ± 0.022	0.147 ± 0.039
NearMiss2	0.463 ± 0.019	0.229 ± 0.047	0.078 ± 0.017	0.184 ± 0.016	0.217 ± 0.020	0.325 ± 0.017	0.358 ± 0.053	0.613 ± 0.111
NearMiss3	0.912 ± 0.013	0.153 ± 0.046	0.252 ± 0.102	0.153 ± 0.014	0.173 ± 0.035	0.240 ± 0.024	0.217 ± 0.072	0.299 ± 0.126
randunder	0.920 ± 0.006	0.147 ± 0.020	0.125 ± 0.020	0.197 ± 0.013	0.191 ± 0.012	0.239 ± 0.012	0.138 ± 0.020	0.154 ± 0.033
Naïve Bayes: Recall								
	classic	**hitech**	**k1b**	**la1**	**la2**	**ohscal**	**reviews**	**sports**
no_resampling	0.394 ± 0.009	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.013 ± 0.003	0.000 ± 0.000	0.003 ± 0.007
GAIOTE	0.983 ± 0.004	0.736 ± 0.071	1.000 ± 0.000	0.824 ± 0.023	0.840 ± 0.030	0.884 ± 0.016	0.860 ± 0.037	0.948 ± 0.023
nonnGAIOTE	0.983 ± 0.005	0.760 ± 0.063	1.000 ± 0.000	0.846 ± 0.024	0.860 ± 0.031	0.893 ± 0.013	0.879 ± 0.037	0.957 ± 0.029
nonnSMOTE	0.983 ± 0.004	0.738 ± 0.063	0.997 ± 0.011	0.875 ± 0.021	0.902 ± 0.024	0.877 ± 0.015	0.837 ± 0.044	0.926 ± 0.037
NBOTE	0.989 ± 0.003	0.762 ± 0.064	1.000 ± 0.000	0.846 ± 0.031	0.868 ± 0.025	0.896 ± 0.016	0.828 ± 0.045	0.954 ± 0.025
generative	0.960 ± 0.007	0.428 ± 0.046	0.227 ± 0.064	0.251 ± 0.041	0.388 ± 0.039	0.842 ± 0.018	0.631 ± 0.045	0.867 ± 0.043
pertNBOTE	0.986 ± 0.004	0.726 ± 0.058	0.987 ± 0.023	0.758 ± 0.028	0.786 ± 0.027	0.890 ± 0.016	0.825 ± 0.047	0.970 ± 0.017
pertrand	0.979 ± 0.004	0.716 ± 0.063	0.993 ± 0.014	0.760 ± 0.037	0.773 ± 0.020	0.889 ± 0.015	0.838 ± 0.047	0.961 ± 0.025
randover	0.984 ± 0.005	0.748 ± 0.070	1.000 ± 0.000	0.839 ± 0.033	0.857 ± 0.029	0.891 ± 0.016	0.859 ± 0.036	0.956 ± 0.023
SMOTE	0.982 ± 0.005	0.740 ± 0.065	0.997 ± 0.011	0.851 ± 0.021	0.869 ± 0.030	0.871 ± 0.018	0.826 ± 0.041	0.920 ± 0.045
SMOTE2	0.983 ± 0.004	0.736 ± 0.068	0.997 ± 0.011	0.850 ± 0.028	0.877 ± 0.030	0.872 ± 0.015	0.825 ± 0.042	0.921 ± 0.041
Distant1	1.000 ± 0.000	0.998 ± 0.005	1.000 ± 0.000	0.949 ± 0.015	0.975 ± 0.010	0.993 ± 0.004	1.000 ± 0.000	1.000 ± 0.000
Distant2	0.968 ± 0.004	1.000 ± 0.000	1.000 ± 0.000	0.971 ± 0.015	0.982 ± 0.007	0.969 ± 0.012	0.997 ± 0.006	0.993 ± 0.008
Distant3	0.998 ± 0.002	1.000 ± 0.000	1.000 ± 0.000	0.978 ± 0.010	0.979 ± 0.007	0.991 ± 0.005	0.999 ± 0.005	0.997 ± 0.007
NearMiss1	0.989 ± 0.003	0.502 ± 0.097	0.650 ± 0.088	0.957 ± 0.015	0.975 ± 0.014	0.930 ± 0.014	0.997 ± 0.006	0.982 ± 0.018
NearMiss2	0.991 ± 0.004	0.555 ± 0.113	1.000 ± 0.000	0.900 ± 0.029	0.889 ± 0.028	0.944 ± 0.018	0.922 ± 0.043	0.980 ± 0.015
NearMiss3	0.987 ± 0.004	0.421 ± 0.081	0.623 ± 0.163	0.865 ± 0.029	0.894 ± 0.042	0.754 ± 0.020	0.990 ± 0.012	0.967 ± 0.015
randunder	0.994 ± 0.002	0.924 ± 0.045	1.000 ± 0.000	0.938 ± 0.025	0.952 ± 0.019	0.975 ± 0.011	0.996 ± 0.007	0.990 ± 0.008
Naïve Bayes: F-measure								
	classic	**hitech**	**k1b**	**la1**	**la2**	**ohscal**	**reviews**	**sports**
no_resampling	0.565 ± 0.010	NaN	NaN	NaN	NaN	0.026 ± 0.006	NaN	NaN
GAIOTE	0.978 ± 0.004	0.412 ± 0.034	0.545 ± 0.027	0.492 ± 0.026	0.508 ± 0.023	0.564 ± 0.011	0.746 ± 0.024	0.903 ± 0.026
nonnGAIOTE	0.977 ± 0.004	0.428 ± 0.033	0.520 ± 0.026	0.452 ± 0.020	0.471 ± 0.026	0.543 ± 0.010	0.718 ± 0.025	0.867 ± 0.038
nonnSMOTE	0.974 ± 0.004	0.429 ± 0.038	0.544 ± 0.026	0.415 ± 0.018	0.437 ± 0.025	0.547 ± 0.010	0.722 ± 0.024	0.894 ± 0.019
NBOTE	0.976 ± 0.004	0.423 ± 0.033	0.529 ± 0.027	0.484 ± 0.025	0.495 ± 0.028	0.543 ± 0.010	0.729 ± 0.026	0.873 ± 0.035
generative	0.972 ± 0.005	0.529 ± 0.029	0.355 ± 0.084	0.384 ± 0.052	0.530 ± 0.033	0.588 ± 0.011	0.764 ± 0.035	0.864 ± 0.032
pertNBOTE	0.978 ± 0.004	0.472 ± 0.040	0.604 ± 0.044	0.551 ± 0.035	0.559 ± 0.030	0.548 ± 0.010	0.793 ± 0.023	0.832 ± 0.041
pertrand	0.977 ± 0.004	0.472 ± 0.044	0.608 ± 0.038	0.544 ± 0.021	0.545 ± 0.028	0.550 ± 0.010	0.790 ± 0.019	0.830 ± 0.042
randover	0.977 ± 0.004	0.432 ± 0.035	0.531 ± 0.029	0.470 ± 0.022	0.483 ± 0.026	0.546 ± 0.011	0.721 ± 0.020	0.874 ± 0.031
SMOTE	0.974 ± 0.003	0.422 ± 0.035	0.564 ± 0.025	0.455 ± 0.022	0.475 ± 0.023	0.571 ± 0.010	0.747 ± 0.029	0.913 ± 0.020
SMOTE2	0.975 ± 0.005	0.417 ± 0.034	0.562 ± 0.027	0.450 ± 0.023	0.475 ± 0.023	0.569 ± 0.011	0.742 ± 0.031	0.914 ± 0.020
Distant1	0.540 ± 0.031	0.147 ± 0.009	0.193 ± 0.015	0.267 ± 0.017	0.252 ± 0.015	0.230 ± 0.006	0.086 ± 0.001	0.047 ± 0.004
Distant2	0.804 ± 0.011	0.149 ± 0.005	0.183 ± 0.018	0.257 ± 0.016	0.233 ± 0.009	0.285 ± 0.008	0.123 ± 0.006	0.050 ± 0.006
Distant3	0.351 ± 0.030	0.102 ± 0.002	0.114 ± 0.012	0.232 ± 0.017	0.218 ± 0.012	0.176 ± 0.012	0.092 ± 0.005	0.033 ± 0.001
NearMiss1	0.937 ± 0.010	0.115 ± 0.024	0.419 ± 0.091	0.228 ± 0.014	0.217 ± 0.010	0.290 ± 0.005	0.224 ± 0.034	0.254 ± 0.057
NearMiss2	0.631 ± 0.018	0.321 ± 0.056	0.145 ± 0.028	0.305 ± 0.021	0.348 ± 0.025	0.483 ± 0.019	0.512 ± 0.048	0.749 ± 0.082
NearMiss3	0.948 ± 0.007	0.222 ± 0.056	0.337 ± 0.084	0.259 ± 0.019	0.288 ± 0.049	0.363 ± 0.029	0.350 ± 0.097	0.445 ± 0.142
randunder	0.955 ± 0.003	0.253 ± 0.029	0.222 ± 0.031	0.325 ± 0.018	0.318 ± 0.016	0.383 ± 0.015	0.243 ± 0.031	0.266 ± 0.049

Table 7: Results for multinomial naïve Bayes

NaN used to denote case where there are no true positives and no true negatives

knn(k=1): Precision								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.971 ± 0.006	0.554 ± 0.046	0.594 ± 0.108	0.469 ± 0.045	0.546 ± 0.034	0.554 ± 0.025	0.811 ± 0.035	0.842 ± 0.041
GAIOTE	0.968 ± 0.007	0.450 ± 0.067	0.475 ± 0.059	0.406 ± 0.040	0.463 ± 0.040	0.447 ± 0.022	0.654 ± 0.056	0.763 ± 0.050
nonnGAIOTE	0.960 ± 0.008	0.428 ± 0.060	0.475 ± 0.055	0.378 ± 0.034	0.427 ± 0.037	0.411 ± 0.021	0.610 ± 0.053	0.730 ± 0.050
nonnSMOTE	0.976 ± 0.007	0.517 ± 0.063	0.529 ± 0.065	0.455 ± 0.040	0.526 ± 0.053	0.542 ± 0.024	0.730 ± 0.058	0.841 ± 0.036
NBOTE	0.974 ± 0.007	0.555 ± 0.056	0.600 ± 0.091	0.469 ± 0.043	0.548 ± 0.053	0.551 ± 0.026	0.765 ± 0.055	0.844 ± 0.034
generative	0.973 ± 0.006	0.521 ± 0.059	0.580 ± 0.089	0.476 ± 0.038	0.572 ± 0.056	0.558 ± 0.017	0.738 ± 0.047	0.880 ± 0.037
pertNBOTE	0.964 ± 0.009	0.429 ± 0.065	0.535 ± 0.063	0.412 ± 0.037	0.494 ± 0.038	0.449 ± 0.019	0.539 ± 0.052	0.789 ± 0.025
pertrand	0.965 ± 0.008	0.436 ± 0.061	0.521 ± 0.067	0.417 ± 0.031	0.496 ± 0.051	0.458 ± 0.016	0.570 ± 0.061	0.780 ± 0.029
randover	0.974 ± 0.008	0.558 ± 0.066	0.591 ± 0.077	0.470 ± 0.047	0.544 ± 0.054	0.555 ± 0.021	0.772 ± 0.057	0.842 ± 0.042
SMOTE	0.972 ± 0.007	0.512 ± 0.064	0.539 ± 0.062	0.438 ± 0.031	0.517 ± 0.049	0.523 ± 0.024	0.757 ± 0.047	0.828 ± 0.045
SMOTE2	0.968 ± 0.007	0.438 ± 0.057	0.465 ± 0.054	0.410 ± 0.030	0.455 ± 0.039	0.450 ± 0.022	0.644 ± 0.054	0.772 ± 0.052
Distant1	0.676 ± 0.023	0.090 ± 0.007	0.103 ± 0.006	0.136 ± 0.007	0.144 ± 0.007	0.134 ± 0.005	0.048 ± 0.001	0.041 ± 0.006
Distant2	0.509 ± 0.019	0.094 ± 0.008	0.110 ± 0.009	0.158 ± 0.010	0.159 ± 0.007	0.166 ± 0.004	0.087 ± 0.004	0.034 ± 0.002
Distant3	0.190 ± 0.006	0.064 ± 0.004	0.045 ± 0.007	0.117 ± 0.004	0.122 ± 0.003	0.090 ± 0.003	0.066 ± 0.006	0.023 ± 0.002
NearMiss1	0.864 ± 0.012	0.078 ± 0.010	0.083 ± 0.013	0.152 ± 0.009	0.167 ± 0.009	0.149 ± 0.004	0.128 ± 0.014	0.097 ± 0.018
NearMiss2	0.382 ± 0.011	0.119 ± 0.010	0.077 ± 0.009	0.151 ± 0.008	0.174 ± 0.010	0.179 ± 0.006	0.135 ± 0.013	0.127 ± 0.026
NearMiss3	0.858 ± 0.011	0.100 ± 0.010	0.065 ± 0.009	0.135 ± 0.010	0.165 ± 0.013	0.125 ± 0.007	0.155 ± 0.015	0.103 ± 0.018
randunder	0.869 ± 0.012	0.137 ± 0.017	0.131 ± 0.009	0.178 ± 0.010	0.195 ± 0.011	0.193 ± 0.006	0.156 ± 0.012	0.121 ± 0.021
knn(k=1): Recall								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.859 ± 0.015	0.464 ± 0.053	0.520 ± 0.098	0.445 ± 0.051	0.553 ± 0.043	0.554 ± 0.032	0.796 ± 0.034	0.846 ± 0.065
GAIOTE	0.885 ± 0.015	0.538 ± 0.061	0.600 ± 0.101	0.531 ± 0.058	0.588 ± 0.041	0.631 ± 0.025	0.688 ± 0.046	0.802 ± 0.071
nonnGAIOTE	0.880 ± 0.012	0.538 ± 0.067	0.607 ± 0.102	0.535 ± 0.058	0.594 ± 0.050	0.621 ± 0.026	0.681 ± 0.054	0.828 ± 0.064
nonnSMOTE	0.846 ± 0.016	0.484 ± 0.062	0.607 ± 0.097	0.474 ± 0.058	0.548 ± 0.047	0.527 ± 0.028	0.637 ± 0.055	0.774 ± 0.069
NBOTE	0.842 ± 0.018	0.450 ± 0.056	0.543 ± 0.101	0.460 ± 0.055	0.540 ± 0.049	0.509 ± 0.029	0.593 ± 0.065	0.770 ± 0.060
generative	0.841 ± 0.013	0.453 ± 0.055	0.503 ± 0.079	0.447 ± 0.046	0.520 ± 0.047	0.496 ± 0.024	0.662 ± 0.038	0.743 ± 0.070
pertNBOTE	0.859 ± 0.017	0.481 ± 0.058	0.533 ± 0.087	0.491 ± 0.059	0.547 ± 0.047	0.576 ± 0.021	0.622 ± 0.054	0.780 ± 0.073
pertrand	0.859 ± 0.020	0.498 ± 0.059	0.540 ± 0.087	0.487 ± 0.054	0.548 ± 0.052	0.580 ± 0.024	0.628 ± 0.052	0.774 ± 0.069
randover	0.842 ± 0.018	0.452 ± 0.051	0.540 ± 0.098	0.459 ± 0.054	0.529 ± 0.043	0.511 ± 0.031	0.607 ± 0.055	0.764 ± 0.061
SMOTE	0.874 ± 0.013	0.502 ± 0.068	0.593 ± 0.089	0.506 ± 0.062	0.568 ± 0.044	0.562 ± 0.032	0.681 ± 0.049	0.780 ± 0.068
SMOTE2	0.885 ± 0.015	0.540 ± 0.067	0.603 ± 0.095	0.554 ± 0.063	0.596 ± 0.042	0.617 ± 0.017	0.706 ± 0.047	0.795 ± 0.062
Distant1	0.979 ± 0.005	0.943 ± 0.018	0.990 ± 0.022	0.859 ± 0.032	0.919 ± 0.016	0.948 ± 0.012	0.997 ± 0.006	0.997 ± 0.007
Distant2	0.974 ± 0.005	0.919 ± 0.018	0.990 ± 0.022	0.811 ± 0.035	0.903 ± 0.027	0.906 ± 0.012	0.974 ± 0.012	0.977 ± 0.025
Distant3	0.999 ± 0.001	0.964 ± 0.025	0.990 ± 0.022	0.929 ± 0.027	0.922 ± 0.023	0.977 ± 0.007	0.984 ± 0.013	0.985 ± 0.018
NearMiss1	0.929 ± 0.012	0.709 ± 0.057	0.633 ± 0.077	0.768 ± 0.041	0.828 ± 0.049	0.825 ± 0.016	0.974 ± 0.018	0.931 ± 0.055
NearMiss2	0.978 ± 0.005	0.736 ± 0.057	0.913 ± 0.088	0.847 ± 0.018	0.831 ± 0.032	0.818 ± 0.013	0.974 ± 0.012	0.944 ± 0.046
NearMiss3	0.917 ± 0.013	0.664 ± 0.056	0.687 ± 0.077	0.733 ± 0.039	0.782 ± 0.037	0.784 ± 0.020	0.946 ± 0.022	0.928 ± 0.048
randunder	0.957 ± 0.008	0.838 ± 0.066	0.973 ± 0.038	0.801 ± 0.035	0.810 ± 0.028	0.895 ± 0.021	0.978 ± 0.021	0.975 ± 0.025
knn(k=1): F-measure								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.911 ± 0.008	0.503 ± 0.042	0.546 ± 0.076	0.456 ± 0.045	0.549 ± 0.032	0.554 ± 0.021	0.803 ± 0.023	0.842 ± 0.034
GAIOTE	0.925 ± 0.009	0.489 ± 0.060	0.528 ± 0.069	0.459 ± 0.045	0.517 ± 0.037	0.523 ± 0.020	0.669 ± 0.041	0.780 ± 0.045
nonnGAIOTE	0.918 ± 0.006	0.476 ± 0.060	0.531 ± 0.069	0.443 ± 0.040	0.496 ± 0.040	0.495 ± 0.019	0.643 ± 0.048	0.774 ± 0.036
nonnSMOTE	0.906 ± 0.009	0.499 ± 0.056	0.563 ± 0.071	0.464 ± 0.046	0.536 ± 0.045	0.534 ± 0.020	0.679 ± 0.049	0.804 ± 0.037
NBOTE	0.903 ± 0.011	0.495 ± 0.050	0.564 ± 0.079	0.464 ± 0.046	0.543 ± 0.043	0.528 ± 0.017	0.666 ± 0.049	0.804 ± 0.033
generative	0.902 ± 0.006	0.484 ± 0.051	0.535 ± 0.069	0.461 ± 0.039	0.544 ± 0.046	0.525 ± 0.014	0.697 ± 0.029	0.803 ± 0.041
pertNBOTE	0.908 ± 0.008	0.453 ± 0.059	0.531 ± 0.067	0.448 ± 0.045	0.518 ± 0.036	0.504 ± 0.017	0.577 ± 0.047	0.783 ± 0.037
pertrand	0.909 ± 0.011	0.464 ± 0.056	0.525 ± 0.057	0.449 ± 0.040	0.520 ± 0.048	0.512 ± 0.016	0.596 ± 0.045	0.774 ± 0.034
randover	0.903 ± 0.010	0.498 ± 0.048	0.558 ± 0.069	0.464 ± 0.049	0.535 ± 0.043	0.531 ± 0.018	0.678 ± 0.045	0.799 ± 0.036
SMOTE	0.920 ± 0.007	0.505 ± 0.059	0.563 ± 0.067	0.468 ± 0.043	0.540 ± 0.041	0.541 ± 0.023	0.716 ± 0.040	0.801 ± 0.039
SMOTE2	0.925 ± 0.008	0.482 ± 0.056	0.523 ± 0.061	0.471 ± 0.040	0.515 ± 0.038	0.520 ± 0.017	0.673 ± 0.048	0.781 ± 0.037
Distant1	0.800 ± 0.015	0.164 ± 0.011	0.186 ± 0.010	0.234 ± 0.011	0.250 ± 0.010	0.235 ± 0.007	0.092 ± 0.002	0.079 ± 0.010
Distant2	0.668 ± 0.017	0.170 ± 0.013	0.198 ± 0.014	0.264 ± 0.015	0.271 ± 0.010	0.281 ± 0.006	0.161 ± 0.007	0.067 ± 0.004
Distant3	0.319 ± 0.008	0.119 ± 0.007	0.086 ± 0.013	0.208 ± 0.007	0.216 ± 0.005	0.165 ± 0.005	0.124 ± 0.011	0.046 ± 0.004
NearMiss1	0.895 ± 0.010	0.140 ± 0.017	0.147 ± 0.021	0.253 ± 0.013	0.278 ± 0.015	0.253 ± 0.006	0.225 ± 0.022	0.175 ± 0.029
NearMiss2	0.549 ± 0.011	0.205 ± 0.017	0.142 ± 0.015	0.256 ± 0.011	0.287 ± 0.014	0.294 ± 0.008	0.236 ± 0.020	0.222 ± 0.038
NearMiss3	0.886 ± 0.009	0.174 ± 0.015	0.119 ± 0.015	0.228 ± 0.015	0.273 ± 0.019	0.215 ± 0.011	0.266 ± 0.022	0.185 ± 0.029
randunder	0.911 ± 0.006	0.236 ± 0.026	0.231 ± 0.014	0.291 ± 0.014	0.314 ± 0.016	0.317 ± 0.008	0.268 ± 0.017	0.215 ± 0.033

Table 8: Results for k nearest neighbors, with k=1

knn(k=5): Precision								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.986 ± 0.004	0.694 ± 0.102	0.668 ± 0.160	0.644 ± 0.063	0.727 ± 0.056	0.768 ± 0.027	0.970 ± 0.028	0.942 ± 0.037
GAIOTE	0.957 ± 0.008	0.342 ± 0.034	0.397 ± 0.031	0.326 ± 0.023	0.379 ± 0.032	0.362 ± 0.013	0.518 ± 0.028	0.645 ± 0.034
nonnGAIOTE	0.949 ± 0.010	0.329 ± 0.020	0.405 ± 0.032	0.320 ± 0.021	0.365 ± 0.023	0.342 ± 0.015	0.491 ± 0.048	0.618 ± 0.046
nonnSMOTE	0.955 ± 0.010	0.371 ± 0.020	0.440 ± 0.044	0.326 ± 0.027	0.393 ± 0.029	0.387 ± 0.020	0.573 ± 0.035	0.700 ± 0.035
NBOTE	0.948 ± 0.009	0.384 ± 0.027	0.485 ± 0.049	0.323 ± 0.021	0.394 ± 0.032	0.389 ± 0.017	0.589 ± 0.047	0.709 ± 0.051
generative	0.983 ± 0.003	0.509 ± 0.060	0.694 ± 0.137	0.602 ± 0.076	0.741 ± 0.064	0.697 ± 0.018	0.810 ± 0.050	0.968 ± 0.016
pertNBOTE	0.958 ± 0.009	0.354 ± 0.034	0.490 ± 0.047	0.352 ± 0.022	0.427 ± 0.035	0.350 ± 0.014	0.537 ± 0.036	0.687 ± 0.024
pertrand	0.958 ± 0.009	0.356 ± 0.035	0.499 ± 0.048	0.359 ± 0.023	0.424 ± 0.035	0.356 ± 0.016	0.541 ± 0.036	0.694 ± 0.029
randover	0.949 ± 0.009	0.381 ± 0.028	0.488 ± 0.046	0.324 ± 0.024	0.393 ± 0.031	0.390 ± 0.016	0.606 ± 0.053	0.702 ± 0.045
SMOTE	0.951 ± 0.008	0.357 ± 0.019	0.433 ± 0.042	0.317 ± 0.019	0.383 ± 0.028	0.379 ± 0.017	0.561 ± 0.038	0.688 ± 0.039
SMOTE2	0.956 ± 0.009	0.331 ± 0.023	0.396 ± 0.040	0.315 ± 0.017	0.374 ± 0.030	0.360 ± 0.016	0.504 ± 0.037	0.654 ± 0.030
Distant1	0.695 ± 0.023	0.085 ± 0.007	0.119 ± 0.006	0.140 ± 0.007	0.145 ± 0.008	0.136 ± 0.004	0.047 ± 0.001	0.036 ± 0.004
Distant2	0.591 ± 0.017	0.088 ± 0.005	0.116 ± 0.008	0.164 ± 0.005	0.159 ± 0.009	0.181 ± 0.007	0.080 ± 0.004	0.034 ± 0.002
Distant3	0.203 ± 0.009	0.059 ± 0.004	0.057 ± 0.011	0.127 ± 0.004	0.133 ± 0.008	0.101 ± 0.005	0.058 ± 0.005	0.022 ± 0.002
NearMiss1	0.917 ± 0.011	0.076 ± 0.016	0.156 ± 0.045	0.162 ± 0.008	0.167 ± 0.008	0.176 ± 0.006	0.151 ± 0.020	0.142 ± 0.030
NearMiss2	0.414 ± 0.012	0.150 ± 0.022	0.086 ± 0.014	0.158 ± 0.010	0.186 ± 0.012	0.226 ± 0.006	0.181 ± 0.023	0.244 ± 0.046
NearMiss3	0.909 ± 0.008	0.113 ± 0.024	0.107 ± 0.024	0.142 ± 0.013	0.180 ± 0.018	0.138 ± 0.013	0.210 ± 0.028	0.172 ± 0.039
randunder	0.919 ± 0.012	0.144 ± 0.016	0.132 ± 0.020	0.193 ± 0.007	0.205 ± 0.008	0.219 ± 0.011	0.166 ± 0.018	0.139 ± 0.028
knn(k=5): Recall								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.871 ± 0.014	0.334 ± 0.046	0.353 ± 0.059	0.339 ± 0.042	0.465 ± 0.029	0.492 ± 0.027	0.800 ± 0.042	0.856 ± 0.066
GAIOTE	0.944 ± 0.011	0.645 ± 0.068	0.817 ± 0.067	0.657 ± 0.059	0.715 ± 0.035	0.763 ± 0.022	0.824 ± 0.035	0.862 ± 0.050
nonnGAIOTE	0.936 ± 0.011	0.633 ± 0.057	0.833 ± 0.047	0.668 ± 0.047	0.708 ± 0.031	0.762 ± 0.022	0.813 ± 0.039	0.872 ± 0.047
nonnSMOTE	0.930 ± 0.015	0.629 ± 0.054	0.817 ± 0.065	0.645 ± 0.061	0.706 ± 0.040	0.704 ± 0.026	0.790 ± 0.040	0.849 ± 0.056
NBOTE	0.937 ± 0.013	0.609 ± 0.059	0.780 ± 0.063	0.638 ± 0.059	0.707 ± 0.034	0.727 ± 0.027	0.776 ± 0.041	0.838 ± 0.071
generative	0.849 ± 0.014	0.341 ± 0.058	0.353 ± 0.045	0.317 ± 0.034	0.433 ± 0.039	0.425 ± 0.021	0.666 ± 0.027	0.738 ± 0.071
pertNBOTE	0.926 ± 0.013	0.612 ± 0.053	0.767 ± 0.075	0.588 ± 0.048	0.658 ± 0.036	0.737 ± 0.021	0.771 ± 0.048	0.844 ± 0.067
pertrand	0.928 ± 0.013	0.600 ± 0.062	0.770 ± 0.071	0.589 ± 0.045	0.658 ± 0.037	0.736 ± 0.024	0.768 ± 0.035	0.851 ± 0.067
randover	0.941 ± 0.014	0.602 ± 0.054	0.783 ± 0.057	0.632 ± 0.064	0.706 ± 0.039	0.723 ± 0.026	0.785 ± 0.046	0.843 ± 0.070
SMOTE	0.947 ± 0.009	0.653 ± 0.057	0.827 ± 0.056	0.664 ± 0.060	0.719 ± 0.026	0.738 ± 0.022	0.821 ± 0.035	0.843 ± 0.061
SMOTE2	0.949 ± 0.012	0.657 ± 0.050	0.823 ± 0.047	0.676 ± 0.048	0.719 ± 0.029	0.758 ± 0.021	0.834 ± 0.037	0.856 ± 0.052
Distant1	0.989 ± 0.004	0.971 ± 0.014	0.997 ± 0.011	0.907 ± 0.021	0.957 ± 0.011	0.978 ± 0.009	1.000 ± 0.000	0.998 ± 0.005
Distant2	0.981 ± 0.005	0.962 ± 0.023	0.990 ± 0.016	0.885 ± 0.028	0.937 ± 0.020	0.943 ± 0.009	0.985 ± 0.014	0.980 ± 0.022
Distant3	0.998 ± 0.002	0.979 ± 0.024	0.983 ± 0.024	0.916 ± 0.031	0.933 ± 0.024	0.979 ± 0.009	0.991 ± 0.012	0.987 ± 0.015
NearMiss1	0.954 ± 0.014	0.624 ± 0.078	0.490 ± 0.072	0.829 ± 0.043	0.873 ± 0.034	0.877 ± 0.021	0.993 ± 0.008	0.961 ± 0.018
NearMiss2	0.984 ± 0.007	0.709 ± 0.070	0.977 ± 0.052	0.880 ± 0.025	0.857 ± 0.038	0.892 ± 0.017	0.979 ± 0.012	0.975 ± 0.019
NearMiss3	0.890 ± 0.014	0.521 ± 0.072	0.457 ± 0.112	0.685 ± 0.054	0.737 ± 0.028	0.667 ± 0.040	0.959 ± 0.024	0.895 ± 0.038
randunder	0.977 ± 0.009	0.852 ± 0.059	0.997 ± 0.011	0.865 ± 0.038	0.883 ± 0.031	0.945 ± 0.007	0.991 ± 0.012	0.990 ± 0.011
knn(k=5): F-measure								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.925 ± 0.008	0.449 ± 0.050	0.458 ± 0.074	0.443 ± 0.048	0.565 ± 0.024	0.600 ± 0.022	0.876 ± 0.028	0.895 ± 0.037
GAIOTE	0.950 ± 0.007	0.445 ± 0.038	0.534 ± 0.039	0.436 ± 0.032	0.495 ± 0.031	0.491 ± 0.012	0.636 ± 0.029	0.736 ± 0.020
nonnGAIOTE	0.942 ± 0.005	0.432 ± 0.025	0.545 ± 0.036	0.433 ± 0.028	0.481 ± 0.025	0.472 ± 0.014	0.612 ± 0.046	0.721 ± 0.026
nonnSMOTE	0.942 ± 0.007	0.466 ± 0.026	0.571 ± 0.052	0.433 ± 0.035	0.504 ± 0.030	0.499 ± 0.020	0.664 ± 0.035	0.766 ± 0.023
NBOTE	0.942 ± 0.007	0.470 ± 0.031	0.597 ± 0.052	0.429 ± 0.030	0.506 ± 0.032	0.507 ± 0.017	0.670 ± 0.044	0.765 ± 0.037
generative	0.911 ± 0.007	0.406 ± 0.049	0.464 ± 0.054	0.415 ± 0.045	0.545 ± 0.031	0.527 ± 0.016	0.730 ± 0.028	0.835 ± 0.047
pertNBOTE	0.942 ± 0.007	0.448 ± 0.037	0.597 ± 0.053	0.440 ± 0.029	0.517 ± 0.033	0.474 ± 0.014	0.633 ± 0.036	0.755 ± 0.026
pertrand	0.943 ± 0.007	0.446 ± 0.041	0.605 ± 0.053	0.445 ± 0.027	0.515 ± 0.033	0.480 ± 0.016	0.634 ± 0.029	0.763 ± 0.025
randover	0.945 ± 0.008	0.466 ± 0.032	0.600 ± 0.049	0.428 ± 0.035	0.505 ± 0.032	0.507 ± 0.016	0.684 ± 0.048	0.763 ± 0.031
SMOTE	0.949 ± 0.005	0.461 ± 0.025	0.568 ± 0.048	0.429 ± 0.028	0.500 ± 0.028	0.501 ± 0.017	0.666 ± 0.037	0.755 ± 0.020
SMOTE2	0.952 ± 0.008	0.440 ± 0.025	0.534 ± 0.042	0.430 ± 0.025	0.492 ± 0.029	0.488 ± 0.017	0.628 ± 0.037	0.740 ± 0.024
Distant1	0.816 ± 0.016	0.156 ± 0.012	0.212 ± 0.010	0.243 ± 0.010	0.252 ± 0.012	0.239 ± 0.006	0.089 ± 0.001	0.069 ± 0.008
Distant2	0.737 ± 0.014	0.161 ± 0.009	0.208 ± 0.014	0.276 ± 0.007	0.272 ± 0.014	0.304 ± 0.010	0.148 ± 0.006	0.066 ± 0.005
Distant3	0.338 ± 0.012	0.111 ± 0.008	0.107 ± 0.019	0.224 ± 0.006	0.233 ± 0.012	0.182 ± 0.008	0.110 ± 0.009	0.042 ± 0.003
NearMiss1	0.935 ± 0.009	0.136 ± 0.027	0.235 ± 0.058	0.271 ± 0.012	0.281 ± 0.012	0.293 ± 0.009	0.261 ± 0.030	0.246 ± 0.044
NearMiss2	0.583 ± 0.012	0.247 ± 0.033	0.158 ± 0.023	0.268 ± 0.014	0.306 ± 0.017	0.361 ± 0.007	0.304 ± 0.032	0.388 ± 0.058
NearMiss3	0.899 ± 0.009	0.186 ± 0.037	0.172 ± 0.035	0.235 ± 0.020	0.289 ± 0.024	0.228 ± 0.019	0.344 ± 0.038	0.287 ± 0.055
randunder	0.947 ± 0.008	0.246 ± 0.024	0.232 ± 0.031	0.316 ± 0.009	0.333 ± 0.012	0.356 ± 0.014	0.284 ± 0.026	0.243 ± 0.043

Table 9: Results for k nearest neighbor, with k=5

knn(k=15): Precision								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.991 ± 0.003	0.818 ± 0.059	0.641 ± 0.297	0.703 ± 0.068	0.850 ± 0.047	0.849 ± 0.025	0.998 ± 0.006	0.970 ± 0.021
GAIOTE	0.955 ± 0.008	0.269 ± 0.017	0.324 ± 0.023	0.278 ± 0.014	0.311 ± 0.024	0.305 ± 0.012	0.376 ± 0.021	0.543 ± 0.055
nonnGAIOTE	0.954 ± 0.011	0.266 ± 0.012	0.329 ± 0.025	0.280 ± 0.011	0.305 ± 0.019	0.290 ± 0.010	0.346 ± 0.020	0.502 ± 0.045
nonnSMOTE	0.957 ± 0.011	0.256 ± 0.014	0.352 ± 0.028	0.245 ± 0.014	0.275 ± 0.015	0.282 ± 0.008	0.355 ± 0.012	0.561 ± 0.060
NBOTE	0.928 ± 0.012	0.251 ± 0.015	0.360 ± 0.026	0.228 ± 0.017	0.266 ± 0.013	0.264 ± 0.007	0.366 ± 0.021	0.558 ± 0.060
generative	0.985 ± 0.005	0.381 ± 0.054	0.774 ± 0.156	0.679 ± 0.081	0.866 ± 0.030	0.608 ± 0.031	0.733 ± 0.066	0.925 ± 0.019
pertNBOTE	0.955 ± 0.012	0.268 ± 0.015	0.407 ± 0.041	0.294 ± 0.022	0.338 ± 0.027	0.282 ± 0.010	0.420 ± 0.026	0.606 ± 0.050
pertrand	0.968 ± 0.007	0.275 ± 0.015	0.404 ± 0.041	0.308 ± 0.020	0.349 ± 0.037	0.292 ± 0.009	0.417 ± 0.023	0.602 ± 0.063
randover	0.947 ± 0.006	0.251 ± 0.014	0.356 ± 0.026	0.232 ± 0.016	0.264 ± 0.015	0.260 ± 0.007	0.364 ± 0.023	0.548 ± 0.061
SMOTE	0.954 ± 0.008	0.250 ± 0.013	0.336 ± 0.026	0.250 ± 0.015	0.284 ± 0.019	0.290 ± 0.011	0.355 ± 0.017	0.553 ± 0.059
SMOTE2	0.957 ± 0.007	0.256 ± 0.015	0.322 ± 0.021	0.262 ± 0.012	0.295 ± 0.021	0.308 ± 0.012	0.360 ± 0.017	0.535 ± 0.050
Distant1	0.655 ± 0.025	0.080 ± 0.007	0.124 ± 0.013	0.143 ± 0.007	0.142 ± 0.007	0.130 ± 0.004	0.046 ± 0.000	0.029 ± 0.003
Distant2	0.659 ± 0.013	0.082 ± 0.004	0.109 ± 0.015	0.163 ± 0.007	0.153 ± 0.008	0.180 ± 0.007	0.071 ± 0.004	0.031 ± 0.003
Distant3	0.220 ± 0.011	0.055 ± 0.003	0.066 ± 0.009	0.137 ± 0.003	0.137 ± 0.012	0.110 ± 0.008	0.052 ± 0.003	0.020 ± 0.001
NearMiss1	0.923 ± 0.011	0.074 ± 0.018	0.368 ± 0.149	0.160 ± 0.010	0.163 ± 0.005	0.189 ± 0.007	0.159 ± 0.024	0.159 ± 0.044
NearMiss2	0.441 ± 0.012	0.207 ± 0.040	0.083 ± 0.020	0.165 ± 0.010	0.197 ± 0.011	0.247 ± 0.010	0.229 ± 0.032	0.357 ± 0.069
NearMiss3	0.915 ± 0.009	0.151 ± 0.034	0.222 ± 0.084	0.141 ± 0.020	0.198 ± 0.020	0.166 ± 0.022	0.253 ± 0.043	0.250 ± 0.096
randunder	0.924 ± 0.009	0.149 ± 0.020	0.127 ± 0.024	0.197 ± 0.010	0.204 ± 0.010	0.223 ± 0.010	0.147 ± 0.020	0.130 ± 0.024
knn(k=15): Recall								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.874 ± 0.015	0.174 ± 0.057	0.103 ± 0.076	0.229 ± 0.032	0.363 ± 0.031	0.429 ± 0.020	0.776 ± 0.032	0.818 ± 0.047
GAIOTE	0.961 ± 0.008	0.743 ± 0.051	0.960 ± 0.041	0.756 ± 0.027	0.793 ± 0.022	0.851 ± 0.015	0.916 ± 0.024	0.890 ± 0.042
nonnGAIOTE	0.959 ± 0.009	0.719 ± 0.049	0.943 ± 0.035	0.757 ± 0.031	0.767 ± 0.033	0.846 ± 0.020	0.909 ± 0.021	0.893 ± 0.040
nonnSMOTE	0.955 ± 0.012	0.769 ± 0.051	0.957 ± 0.039	0.793 ± 0.038	0.801 ± 0.021	0.836 ± 0.019	0.913 ± 0.022	0.895 ± 0.042
NBOTE	0.968 ± 0.010	0.769 ± 0.048	0.937 ± 0.058	0.802 ± 0.039	0.813 ± 0.023	0.858 ± 0.020	0.916 ± 0.026	0.885 ± 0.043
generative	0.848 ± 0.011	0.260 ± 0.058	0.153 ± 0.061	0.225 ± 0.041	0.333 ± 0.034	0.405 ± 0.022	0.701 ± 0.039	0.713 ± 0.067
pertNBOTE	0.953 ± 0.011	0.712 ± 0.030	0.907 ± 0.054	0.690 ± 0.048	0.726 ± 0.027	0.833 ± 0.018	0.872 ± 0.023	0.880 ± 0.046
pertrand	0.949 ± 0.010	0.716 ± 0.031	0.907 ± 0.052	0.691 ± 0.037	0.730 ± 0.031	0.838 ± 0.020	0.872 ± 0.024	0.880 ± 0.046
randover	0.963 ± 0.009	0.769 ± 0.048	0.937 ± 0.058	0.801 ± 0.034	0.812 ± 0.032	0.864 ± 0.018	0.912 ± 0.023	0.882 ± 0.044
SMOTE	0.968 ± 0.009	0.786 ± 0.052	0.957 ± 0.045	0.793 ± 0.037	0.805 ± 0.020	0.848 ± 0.020	0.931 ± 0.022	0.893 ± 0.043
SMOTE2	0.966 ± 0.008	0.771 ± 0.048	0.960 ± 0.041	0.794 ± 0.031	0.797 ± 0.028	0.847 ± 0.017	0.932 ± 0.024	0.895 ± 0.044
Distant1	0.996 ± 0.003	0.983 ± 0.016	1.000 ± 0.000	0.940 ± 0.020	0.976 ± 0.007	0.990 ± 0.006	1.000 ± 0.000	0.992 ± 0.009
Distant2	0.983 ± 0.003	0.990 ± 0.015	1.000 ± 0.000	0.929 ± 0.016	0.957 ± 0.017	0.965 ± 0.012	0.993 ± 0.008	0.992 ± 0.009
Distant3	0.991 ± 0.005	0.993 ± 0.012	0.973 ± 0.021	0.932 ± 0.022	0.923 ± 0.023	0.973 ± 0.012	0.993 ± 0.010	0.997 ± 0.007
NearMiss1	0.963 ± 0.008	0.488 ± 0.087	0.387 ± 0.085	0.875 ± 0.045	0.915 ± 0.023	0.914 ± 0.013	0.987 ± 0.011	0.957 ± 0.018
NearMiss2	0.986 ± 0.006	0.633 ± 0.084	0.993 ± 0.014	0.905 ± 0.014	0.880 ± 0.033	0.932 ± 0.014	0.976 ± 0.016	0.977 ± 0.018
NearMiss3	0.903 ± 0.012	0.434 ± 0.080	0.327 ± 0.149	0.637 ± 0.072	0.739 ± 0.073	0.619 ± 0.049	0.949 ± 0.024	0.898 ± 0.041
randunder	0.983 ± 0.006	0.900 ± 0.049	1.000 ± 0.000	0.913 ± 0.031	0.918 ± 0.022	0.974 ± 0.008	0.999 ± 0.005	0.992 ± 0.009
knn(k-15): F-measure								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.929 ± 0.009	0.283 ± 0.081	NaN	0.345 ± 0.043	0.508 ± 0.034	0.569 ± 0.017	0.873 ± 0.020	0.886 ± 0.027
GAIOTE	0.958 ± 0.007	0.395 ± 0.021	0.484 ± 0.028	0.406 ± 0.018	0.446 ± 0.025	0.449 ± 0.013	0.532 ± 0.022	0.672 ± 0.035
nonnGAIOTE	0.956 ± 0.006	0.388 ± 0.015	0.488 ± 0.030	0.408 ± 0.014	0.436 ± 0.023	0.431 ± 0.011	0.501 ± 0.022	0.641 ± 0.032
nonnSMOTE	0.956 ± 0.008	0.384 ± 0.021	0.514 ± 0.034	0.375 ± 0.019	0.409 ± 0.018	0.422 ± 0.010	0.511 ± 0.011	0.687 ± 0.036
NBOTE	0.947 ± 0.009	0.379 ± 0.021	0.519 ± 0.034	0.355 ± 0.023	0.401 ± 0.017	0.403 ± 0.008	0.522 ± 0.022	0.682 ± 0.035
generative	0.911 ± 0.007	0.307 ± 0.054	0.252 ± 0.091	0.337 ± 0.054	0.480 ± 0.035	0.486 ± 0.024	0.716 ± 0.049	0.803 ± 0.040
pertNBOTE	0.954 ± 0.006	0.389 ± 0.019	0.561 ± 0.044	0.412 ± 0.028	0.461 ± 0.026	0.422 ± 0.012	0.566 ± 0.022	0.715 ± 0.026
pertrand	0.958 ± 0.005	0.397 ± 0.020	0.558 ± 0.044	0.425 ± 0.023	0.472 ± 0.037	0.433 ± 0.010	0.564 ± 0.023	0.711 ± 0.033
randover	0.955 ± 0.006	0.378 ± 0.020	0.516 ± 0.034	0.360 ± 0.022	0.399 ± 0.020	0.400 ± 0.009	0.520 ± 0.025	0.673 ± 0.039
SMOTE	0.961 ± 0.005	0.380 ± 0.020	0.497 ± 0.032	0.380 ± 0.020	0.420 ± 0.021	0.433 ± 0.012	0.513 ± 0.019	0.680 ± 0.035
SMOTE2	0.961 ± 0.004	0.384 ± 0.021	0.482 ± 0.028	0.394 ± 0.016	0.431 ± 0.024	0.452 ± 0.013	0.519 ± 0.019	0.668 ± 0.033
Distant1	0.790 ± 0.019	0.148 ± 0.012	0.220 ± 0.020	0.248 ± 0.010	0.248 ± 0.011	0.230 ± 0.006	0.088 ± 0.001	0.057 ± 0.005
Distant2	0.789 ± 0.010	0.152 ± 0.006	0.196 ± 0.024	0.277 ± 0.011	0.264 ± 0.013	0.304 ± 0.010	0.132 ± 0.006	0.060 ± 0.006
Distant3	0.360 ± 0.014	0.105 ± 0.005	0.123 ± 0.015	0.239 ± 0.005	0.238 ± 0.018	0.197 ± 0.012	0.100 ± 0.006	0.038 ± 0.002
NearMiss1	0.942 ± 0.006	0.128 ± 0.030	0.359 ± 0.094	0.270 ± 0.014	0.276 ± 0.007	0.313 ± 0.010	0.274 ± 0.035	0.270 ± 0.062
NearMiss2	0.609 ± 0.011	0.310 ± 0.049	0.152 ± 0.033	0.279 ± 0.014	0.322 ± 0.015	0.390 ± 0.012	0.370 ± 0.040	0.519 ± 0.073
NearMiss3	0.909 ± 0.007	0.222 ± 0.044	0.238 ± 0.052	0.231 ± 0.029	0.312 ± 0.024	0.261 ± 0.031	0.398 ± 0.052	0.384 ± 0.117
randunder	0.952 ± 0.003	0.255 ± 0.029	0.225 ± 0.036	0.324 ± 0.013	0.333 ± 0.013	0.362 ± 0.013	0.255 ± 0.031	0.230 ± 0.037

Table 10: Results for k nearest neighbor, with k=15

knn(k=25): Precision								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.992 ± 0.003	0.896 ± 0.083	NaN	0.709 ± 0.085	0.899 ± 0.045	0.872 ± 0.032	1.000 ± 0.000	0.969 ± 0.022
GAIOTE	0.958 ± 0.007	0.246 ± 0.014	0.282 ± 0.023	0.266 ± 0.010	0.295 ± 0.018	0.301 ± 0.013	0.330 ± 0.022	0.468 ± 0.051
nonnGAIOTE	0.959 ± 0.008	0.244 ± 0.010	0.286 ± 0.022	0.269 ± 0.010	0.292 ± 0.019	0.281 ± 0.011	0.296 ± 0.018	0.436 ± 0.037
nonnSMOTE	0.962 ± 0.009	0.210 ± 0.006	0.293 ± 0.022	0.232 ± 0.009	0.259 ± 0.019	0.265 ± 0.009	0.270 ± 0.015	0.457 ± 0.058
NBOTE	0.925 ± 0.014	0.197 ± 0.008	0.292 ± 0.017	0.216 ± 0.012	0.247 ± 0.010	0.230 ± 0.010	0.271 ± 0.019	0.457 ± 0.054
generative	0.985 ± 0.004	0.446 ± 0.075	0.887 ± 0.158	0.736 ± 0.054	0.895 ± 0.034	0.631 ± 0.023	0.765 ± 0.054	0.847 ± 0.030
pertNBOTE	0.959 ± 0.011	0.242 ± 0.010	0.359 ± 0.031	0.291 ± 0.015	0.322 ± 0.021	0.280 ± 0.011	0.367 ± 0.019	0.518 ± 0.044
pertrand	0.970 ± 0.009	0.249 ± 0.010	0.358 ± 0.034	0.305 ± 0.013	0.340 ± 0.021	0.300 ± 0.010	0.360 ± 0.019	0.514 ± 0.045
randover	0.955 ± 0.010	0.198 ± 0.011	0.287 ± 0.016	0.232 ± 0.007	0.257 ± 0.012	0.235 ± 0.008	0.268 ± 0.017	0.449 ± 0.054
SMOTE	0.957 ± 0.006	0.207 ± 0.013	0.280 ± 0.017	0.238 ± 0.013	0.270 ± 0.013	0.279 ± 0.012	0.281 ± 0.019	0.459 ± 0.061
SMOTE2	0.956 ± 0.006	0.223 ± 0.011	0.278 ± 0.017	0.248 ± 0.012	0.277 ± 0.015	0.294 ± 0.013	0.308 ± 0.024	0.466 ± 0.056
Distant1	0.615 ± 0.026	0.077 ± 0.006	0.116 ± 0.013	0.144 ± 0.007	0.140 ± 0.007	0.124 ± 0.004	0.046 ± 0.000	0.026 ± 0.002
Distant2	0.688 ± 0.017	0.080 ± 0.003	0.102 ± 0.014	0.160 ± 0.011	0.150 ± 0.009	0.174 ± 0.007	0.066 ± 0.003	0.028 ± 0.003
Distant3	0.230 ± 0.012	0.054 ± 0.002	0.073 ± 0.007	0.140 ± 0.006	0.139 ± 0.012	0.113 ± 0.008	0.051 ± 0.003	0.018 ± 0.001
NearMiss1	0.917 ± 0.015	0.078 ± 0.015	0.404 ± 0.161	0.155 ± 0.009	0.156 ± 0.007	0.191 ± 0.006	0.159 ± 0.028	0.154 ± 0.043
NearMiss2	0.454 ± 0.011	0.263 ± 0.050	0.077 ± 0.017	0.169 ± 0.010	0.202 ± 0.011	0.252 ± 0.012	0.260 ± 0.034	0.418 ± 0.099
NearMiss3	0.911 ± 0.012	0.188 ± 0.055	0.266 ± 0.069	0.145 ± 0.016	0.209 ± 0.029	0.179 ± 0.029	0.252 ± 0.051	0.280 ± 0.124
randunder	0.923 ± 0.007	0.150 ± 0.022	0.121 ± 0.023	0.191 ± 0.011	0.200 ± 0.015	0.216 ± 0.011	0.132 ± 0.020	0.122 ± 0.021
knn(k=25): Recall								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.867 ± 0.012	0.116 ± 0.044	0.040 ± 0.052	0.193 ± 0.023	0.325 ± 0.036	0.384 ± 0.024	0.759 ± 0.041	0.811 ± 0.045
GAIOTE	0.966 ± 0.008	0.803 ± 0.056	0.983 ± 0.024	0.802 ± 0.023	0.814 ± 0.026	0.883 ± 0.015	0.950 ± 0.025	0.916 ± 0.037
nonnGAIOTE	0.959 ± 0.007	0.764 ± 0.049	0.983 ± 0.018	0.790 ± 0.026	0.798 ± 0.039	0.875 ± 0.017	0.941 ± 0.025	0.915 ± 0.044
nonnSMOTE	0.959 ± 0.008	0.816 ± 0.049	0.983 ± 0.018	0.827 ± 0.027	0.831 ± 0.024	0.875 ± 0.014	0.951 ± 0.017	0.911 ± 0.042
NBOTE	0.974 ± 0.009	0.836 ± 0.055	0.970 ± 0.025	0.841 ± 0.024	0.842 ± 0.024	0.901 ± 0.014	0.954 ± 0.021	0.918 ± 0.047
generative	0.841 ± 0.013	0.257 ± 0.046	0.117 ± 0.067	0.196 ± 0.032	0.295 ± 0.032	0.420 ± 0.021	0.682 ± 0.046	0.679 ± 0.048
pertNBOTE	0.960 ± 0.006	0.764 ± 0.053	0.933 ± 0.052	0.728 ± 0.041	0.754 ± 0.032	0.865 ± 0.011	0.915 ± 0.026	0.898 ± 0.040
pertrand	0.952 ± 0.009	0.752 ± 0.031	0.940 ± 0.056	0.733 ± 0.046	0.759 ± 0.027	0.869 ± 0.018	0.918 ± 0.025	0.895 ± 0.035
randover	0.969 ± 0.007	0.831 ± 0.055	0.967 ± 0.031	0.827 ± 0.027	0.827 ± 0.037	0.899 ± 0.017	0.956 ± 0.020	0.908 ± 0.041
SMOTE	0.972 ± 0.009	0.831 ± 0.066	0.980 ± 0.023	0.835 ± 0.029	0.832 ± 0.021	0.884 ± 0.015	0.954 ± 0.022	0.916 ± 0.041
SMOTE2	0.970 ± 0.007	0.819 ± 0.063	0.983 ± 0.018	0.838 ± 0.021	0.837 ± 0.022	0.879 ± 0.016	0.956 ± 0.021	0.918 ± 0.039
Distant1	0.997 ± 0.002	0.997 ± 0.007	1.000 ± 0.000	0.958 ± 0.014	0.979 ± 0.008	0.992 ± 0.005	1.000 ± 0.000	0.990 ± 0.008
Distant2	0.980 ± 0.003	0.998 ± 0.005	1.000 ± 0.000	0.949 ± 0.011	0.966 ± 0.013	0.969 ± 0.011	1.000 ± 0.000	0.995 ± 0.008
Distant3	0.987 ± 0.006	0.986 ± 0.011	0.967 ± 0.027	0.914 ± 0.023	0.926 ± 0.027	0.969 ± 0.011	0.999 ± 0.005	0.997 ± 0.007
NearMiss1	0.965 ± 0.006	0.438 ± 0.064	0.400 ± 0.119	0.902 ± 0.035	0.937 ± 0.023	0.926 ± 0.009	0.981 ± 0.012	0.948 ± 0.019
NearMiss2	0.986 ± 0.007	0.583 ± 0.088	1.000 ± 0.000	0.915 ± 0.016	0.889 ± 0.027	0.951 ± 0.012	0.981 ± 0.018	0.982 ± 0.016
NearMiss3	0.907 ± 0.009	0.390 ± 0.089	0.347 ± 0.184	0.641 ± 0.060	0.737 ± 0.096	0.577 ± 0.069	0.907 ± 0.044	0.892 ± 0.053
randunder	0.984 ± 0.007	0.914 ± 0.049	1.000 ± 0.000	0.931 ± 0.023	0.937 ± 0.018	0.978 ± 0.010	0.999 ± 0.005	0.992 ± 0.009
knn(k=25): F-measure								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.926 ± 0.007	0.202 ± 0.069	NaN	0.302 ± 0.031	0.476 ± 0.039	0.533 ± 0.025	0.862 ± 0.027	0.883 ± 0.028
GAIOTE	0.962 ± 0.005	0.377 ± 0.018	0.438 ± 0.028	0.399 ± 0.012	0.433 ± 0.020	0.449 ± 0.015	0.489 ± 0.026	0.618 ± 0.040
nonnGAIOTE	0.959 ± 0.004	0.369 ± 0.013	0.442 ± 0.027	0.402 ± 0.012	0.428 ± 0.024	0.425 ± 0.013	0.451 ± 0.023	0.589 ± 0.028
nonnSMOTE	0.961 ± 0.006	0.334 ± 0.011	0.451 ± 0.027	0.362 ± 0.012	0.394 ± 0.024	0.407 ± 0.011	0.420 ± 0.020	0.606 ± 0.043
NBOTE	0.949 ± 0.010	0.319 ± 0.013	0.448 ± 0.020	0.344 ± 0.016	0.381 ± 0.013	0.366 ± 0.013	0.421 ± 0.024	0.607 ± 0.038
generative	0.908 ± 0.008	0.324 ± 0.052	0.200 ± 0.103	0.308 ± 0.042	0.443 ± 0.038	0.504 ± 0.019	0.721 ± 0.047	0.753 ± 0.033
pertNBOTE	0.959 ± 0.004	0.367 ± 0.015	0.518 ± 0.037	0.415 ± 0.021	0.451 ± 0.023	0.423 ± 0.013	0.524 ± 0.020	0.655 ± 0.026
pertrand	0.961 ± 0.003	0.374 ± 0.013	0.518 ± 0.041	0.431 ± 0.017	0.470 ± 0.021	0.446 ± 0.012	0.517 ± 0.023	0.651 ± 0.030
randover	0.962 ± 0.005	0.320 ± 0.018	0.443 ± 0.021	0.362 ± 0.010	0.392 ± 0.016	0.373 ± 0.011	0.418 ± 0.021	0.598 ± 0.040
SMOTE	0.965 ± 0.005	0.331 ± 0.020	0.435 ± 0.022	0.371 ± 0.017	0.408 ± 0.015	0.424 ± 0.013	0.434 ± 0.025	0.608 ± 0.046
SMOTE2	0.963 ± 0.004	0.350 ± 0.018	0.433 ± 0.022	0.382 ± 0.015	0.416 ± 0.017	0.440 ± 0.015	0.466 ± 0.029	0.616 ± 0.043
Distant1	0.760 ± 0.020	0.143 ± 0.010	0.208 ± 0.022	0.251 ± 0.010	0.245 ± 0.011	0.221 ± 0.006	0.087 ± 0.001	0.051 ± 0.004
Distant2	0.808 ± 0.012	0.147 ± 0.005	0.186 ± 0.024	0.274 ± 0.016	0.259 ± 0.013	0.295 ± 0.010	0.123 ± 0.006	0.054 ± 0.006
Distant3	0.373 ± 0.016	0.103 ± 0.004	0.136 ± 0.012	0.243 ± 0.009	0.241 ± 0.019	0.202 ± 0.013	0.097 ± 0.005	0.036 ± 0.002
NearMiss1	0.940 ± 0.006	0.132 ± 0.023	0.373 ± 0.106	0.264 ± 0.014	0.268 ± 0.010	0.316 ± 0.008	0.272 ± 0.041	0.262 ± 0.061
NearMiss2	0.621 ± 0.011	0.359 ± 0.048	0.142 ± 0.029	0.285 ± 0.015	0.329 ± 0.014	0.398 ± 0.014	0.410 ± 0.041	0.580 ± 0.091
NearMiss3	0.909 ± 0.008	0.251 ± 0.066	0.263 ± 0.057	0.236 ± 0.023	0.324 ± 0.033	0.273 ± 0.041	0.391 ± 0.062	0.415 ± 0.144
randunder	0.953 ± 0.004	0.257 ± 0.031	0.216 ± 0.036	0.317 ± 0.016	0.330 ± 0.020	0.354 ± 0.015	0.233 ± 0.031	0.216 ± 0.033

Table 11: Results for k nearest neighbor, with k=25

knn(k=35): Precision								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.992 ± 0.003	0.940 ± 0.078	NaN	0.729 ± 0.102	0.923 ± 0.038	0.886 ± 0.022	1.000 ± 0.000	0.978 ± 0.015
GAIOTE	0.958 ± 0.007	0.235 ± 0.015	0.255 ± 0.021	0.258 ± 0.013	0.288 ± 0.015	0.297 ± 0.012	0.312 ± 0.021	0.418 ± 0.053
nonnGAIOTE	0.959 ± 0.006	0.239 ± 0.011	0.257 ± 0.019	0.264 ± 0.008	0.288 ± 0.016	0.285 ± 0.013	0.280 ± 0.018	0.404 ± 0.035
nonnSMOTE	0.968 ± 0.007	0.200 ± 0.008	0.256 ± 0.018	0.224 ± 0.006	0.255 ± 0.015	0.267 ± 0.009	0.235 ± 0.013	0.389 ± 0.050
NBOTE	0.931 ± 0.010	0.190 ± 0.008	0.242 ± 0.015	0.216 ± 0.009	0.250 ± 0.009	0.233 ± 0.014	0.232 ± 0.017	0.393 ± 0.046
generative	0.984 ± 0.004	0.492 ± 0.053	0.960 ± 0.084	0.770 ± 0.075	0.921 ± 0.036	0.641 ± 0.028	0.831 ± 0.056	0.813 ± 0.034
pertNBOTE	0.962 ± 0.009	0.233 ± 0.012	0.325 ± 0.023	0.292 ± 0.009	0.325 ± 0.020	0.288 ± 0.014	0.355 ± 0.023	0.476 ± 0.039
pertrand	0.973 ± 0.007	0.245 ± 0.016	0.327 ± 0.027	0.305 ± 0.015	0.333 ± 0.023	0.307 ± 0.014	0.348 ± 0.020	0.470 ± 0.047
randover	0.958 ± 0.007	0.193 ± 0.008	0.240 ± 0.015	0.229 ± 0.010	0.255 ± 0.012	0.257 ± 0.008	0.222 ± 0.014	0.387 ± 0.053
SMOTE	0.958 ± 0.006	0.198 ± 0.011	0.252 ± 0.018	0.228 ± 0.007	0.262 ± 0.010	0.275 ± 0.010	0.251 ± 0.016	0.398 ± 0.061
SMOTE2	0.957 ± 0.007	0.209 ± 0.010	0.256 ± 0.018	0.234 ± 0.009	0.267 ± 0.012	0.288 ± 0.013	0.279 ± 0.019	0.418 ± 0.060
Distant1	0.582 ± 0.031	0.076 ± 0.005	0.112 ± 0.012	0.144 ± 0.007	0.139 ± 0.007	0.120 ± 0.004	0.045 ± 0.000	0.025 ± 0.002
Distant2	0.701 ± 0.017	0.077 ± 0.003	0.100 ± 0.012	0.156 ± 0.011	0.147 ± 0.008	0.169 ± 0.007	0.062 ± 0.003	0.026 ± 0.003
Distant3	0.238 ± 0.015	0.054 ± 0.002	0.077 ± 0.007	0.144 ± 0.007	0.141 ± 0.014	0.115 ± 0.008	0.050 ± 0.003	0.018 ± 0.001
NearMiss1	0.917 ± 0.015	0.083 ± 0.016	0.425 ± 0.160	0.148 ± 0.009	0.150 ± 0.007	0.190 ± 0.005	0.157 ± 0.029	0.145 ± 0.041
NearMiss2	0.460 ± 0.011	0.319 ± 0.050	0.074 ± 0.016	0.169 ± 0.009	0.204 ± 0.012	0.252 ± 0.011	0.278 ± 0.033	0.432 ± 0.104
NearMiss3	0.905 ± 0.012	0.219 ± 0.060	0.334 ± 0.171	0.150 ± 0.020	0.217 ± 0.044	0.182 ± 0.035	0.250 ± 0.060	0.290 ± 0.136
randunder	0.922 ± 0.008	0.151 ± 0.025	0.118 ± 0.022	0.186 ± 0.011	0.195 ± 0.014	0.209 ± 0.011	0.126 ± 0.020	0.116 ± 0.018
knn(k=35): Recall								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.849 ± 0.013	0.066 ± 0.035	0.020 ± 0.028	0.174 ± 0.016	0.295 ± 0.025	0.349 ± 0.022	0.728 ± 0.048	0.790 ± 0.053
GAIOTE	0.970 ± 0.008	0.822 ± 0.059	0.990 ± 0.016	0.826 ± 0.023	0.838 ± 0.038	0.895 ± 0.014	0.957 ± 0.021	0.931 ± 0.031
nonnGAIOTE	0.962 ± 0.007	0.797 ± 0.053	0.993 ± 0.014	0.820 ± 0.024	0.823 ± 0.038	0.890 ± 0.014	0.956 ± 0.026	0.936 ± 0.041
nonnSMOTE	0.966 ± 0.008	0.843 ± 0.059	0.990 ± 0.016	0.860 ± 0.026	0.852 ± 0.025	0.891 ± 0.012	0.963 ± 0.017	0.925 ± 0.040
NBOTE	0.979 ± 0.006	0.864 ± 0.055	0.990 ± 0.016	0.861 ± 0.022	0.864 ± 0.027	0.914 ± 0.012	0.963 ± 0.022	0.939 ± 0.037
generative	0.835 ± 0.011	0.278 ± 0.034	0.100 ± 0.057	0.190 ± 0.023	0.284 ± 0.028	0.440 ± 0.024	0.672 ± 0.056	0.672 ± 0.052
pertNBOTE	0.963 ± 0.008	0.788 ± 0.067	0.970 ± 0.019	0.761 ± 0.038	0.782 ± 0.033	0.888 ± 0.013	0.932 ± 0.033	0.911 ± 0.045
pertrand	0.955 ± 0.009	0.779 ± 0.041	0.967 ± 0.027	0.754 ± 0.038	0.773 ± 0.031	0.886 ± 0.016	0.932 ± 0.025	0.913 ± 0.035
randover	0.973 ± 0.006	0.829 ± 0.055	0.990 ± 0.016	0.857 ± 0.027	0.844 ± 0.034	0.909 ± 0.016	0.965 ± 0.020	0.933 ± 0.034
SMOTE	0.976 ± 0.007	0.862 ± 0.059	0.990 ± 0.016	0.860 ± 0.025	0.860 ± 0.026	0.904 ± 0.014	0.962 ± 0.020	0.934 ± 0.034
SMOTE2	0.974 ± 0.007	0.840 ± 0.051	0.990 ± 0.016	0.853 ± 0.019	0.854 ± 0.021	0.899 ± 0.017	0.968 ± 0.018	0.930 ± 0.036
Distant1	0.997 ± 0.002	1.000 ± 0.000	1.000 ± 0.000	0.954 ± 0.009	0.979 ± 0.009	0.993 ± 0.004	1.000 ± 0.000	0.990 ± 0.008
Distant2	0.975 ± 0.004	1.000 ± 0.000	1.000 ± 0.000	0.951 ± 0.013	0.973 ± 0.011	0.972 ± 0.010	1.000 ± 0.000	0.995 ± 0.008
Distant3	0.981 ± 0.008	0.978 ± 0.012	0.977 ± 0.032	0.905 ± 0.025	0.932 ± 0.027	0.971 ± 0.011	0.997 ± 0.006	0.997 ± 0.007
NearMiss1	0.967 ± 0.005	0.419 ± 0.057	0.527 ± 0.167	0.918 ± 0.030	0.948 ± 0.018	0.934 ± 0.011	0.979 ± 0.010	0.936 ± 0.021
NearMiss2	0.986 ± 0.006	0.552 ± 0.097	1.000 ± 0.000	0.915 ± 0.016	0.890 ± 0.027	0.962 ± 0.010	0.976 ± 0.021	0.982 ± 0.014
NearMiss3	0.908 ± 0.008	0.369 ± 0.084	0.430 ± 0.196	0.665 ± 0.083	0.741 ± 0.088	0.530 ± 0.090	0.871 ± 0.048	0.907 ± 0.067
randunder	0.984 ± 0.006	0.928 ± 0.032	1.000 ± 0.000	0.940 ± 0.029	0.946 ± 0.020	0.980 ± 0.008	1.000 ± 0.000	0.990 ± 0.008
knn(k=35): F-measure								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.915 ± 0.008	0.120 ± 0.061	NaN	0.280 ± 0.019	0.447 ± 0.030	0.501 ± 0.024	0.842 ± 0.033	0.873 ± 0.034
GAIOTE	0.964 ± 0.006	0.365 ± 0.019	0.405 ± 0.028	0.393 ± 0.016	0.428 ± 0.019	0.446 ± 0.013	0.470 ± 0.025	0.575 ± 0.045
nonnGAIOTE	0.961 ± 0.004	0.368 ± 0.016	0.408 ± 0.025	0.400 ± 0.011	0.427 ± 0.020	0.432 ± 0.015	0.433 ± 0.022	0.563 ± 0.029
nonnSMOTE	0.967 ± 0.005	0.323 ± 0.013	0.406 ± 0.024	0.356 ± 0.008	0.392 ± 0.018	0.411 ± 0.011	0.377 ± 0.017	0.545 ± 0.042
NBOTE	0.955 ± 0.006	0.311 ± 0.014	0.389 ± 0.020	0.346 ± 0.012	0.387 ± 0.011	0.371 ± 0.017	0.374 ± 0.023	0.552 ± 0.040
generative	0.903 ± 0.008	0.353 ± 0.033	0.176 ± 0.091	0.304 ± 0.031	0.433 ± 0.032	0.522 ± 0.021	0.743 ± 0.055	0.735 ± 0.035
pertNBOTE	0.963 ± 0.004	0.360 ± 0.018	0.486 ± 0.027	0.422 ± 0.014	0.458 ± 0.022	0.434 ± 0.016	0.514 ± 0.025	0.624 ± 0.024
pertrand	0.964 ± 0.005	0.372 ± 0.020	0.488 ± 0.032	0.433 ± 0.016	0.465 ± 0.025	0.456 ± 0.017	0.506 ± 0.023	0.618 ± 0.033
randover	0.965 ± 0.004	0.313 ± 0.013	0.387 ± 0.020	0.361 ± 0.014	0.391 ± 0.016	0.400 ± 0.010	0.361 ± 0.020	0.545 ± 0.047
SMOTE	0.967 ± 0.004	0.321 ± 0.016	0.402 ± 0.024	0.360 ± 0.009	0.401 ± 0.012	0.422 ± 0.012	0.399 ± 0.022	0.555 ± 0.051
SMOTE2	0.966 ± 0.005	0.334 ± 0.014	0.406 ± 0.023	0.367 ± 0.011	0.406 ± 0.013	0.437 ± 0.015	0.433 ± 0.023	0.574 ± 0.049
Distant1	0.734 ± 0.025	0.141 ± 0.009	0.201 ± 0.020	0.250 ± 0.011	0.243 ± 0.011	0.215 ± 0.006	0.087 ± 0.001	0.049 ± 0.003
Distant2	0.816 ± 0.013	0.143 ± 0.005	0.182 ± 0.020	0.268 ± 0.015	0.256 ± 0.012	0.288 ± 0.010	0.117 ± 0.006	0.050 ± 0.006
Distant3	0.382 ± 0.019	0.102 ± 0.003	0.142 ± 0.013	0.248 ± 0.011	0.244 ± 0.021	0.206 ± 0.013	0.096 ± 0.005	0.035 ± 0.002
NearMiss1	0.941 ± 0.006	0.139 ± 0.024	0.432 ± 0.111	0.255 ± 0.014	0.259 ± 0.011	0.316 ± 0.007	0.270 ± 0.044	0.250 ± 0.059
NearMiss2	0.627 ± 0.010	0.401 ± 0.050	0.138 ± 0.027	0.285 ± 0.013	0.332 ± 0.016	0.399 ± 0.013	0.432 ± 0.038	0.594 ± 0.094
NearMiss3	0.907 ± 0.008	0.271 ± 0.064	0.328 ± 0.086	0.245 ± 0.031	0.331 ± 0.048	0.271 ± 0.050	0.385 ± 0.074	0.427 ± 0.159
randunder	0.952 ± 0.004	0.258 ± 0.037	0.211 ± 0.034	0.311 ± 0.015	0.323 ± 0.018	0.345 ± 0.014	0.223 ± 0.032	0.206 ± 0.030

Table 12: Results for k nearest neighbor, with k=35

SVM: Precision								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.999 ± 0.001	0.896 ± 0.075	0.663 ± 0.274	0.811 ± 0.051	0.932 ± 0.034	0.858 ± 0.018	1.000 ± 0.000	0.979 ± 0.016
GAIOTE	0.999 ± 0.001	0.744 ± 0.094	0.561 ± 0.133	0.817 ± 0.043	0.864 ± 0.023	0.751 ± 0.021	0.994 ± 0.010	0.987 ± 0.011
nonnGAIOTE	0.998 ± 0.001	0.718 ± 0.084	0.587 ± 0.116	0.787 ± 0.049	0.850 ± 0.033	0.713 ± 0.020	0.994 ± 0.010	0.986 ± 0.010
nonnSMOTE	0.999 ± 0.001	0.805 ± 0.077	0.532 ± 0.168	0.814 ± 0.036	0.877 ± 0.033	0.812 ± 0.017	0.998 ± 0.007	0.987 ± 0.013
NBOTE	0.999 ± 0.001	0.785 ± 0.086	0.558 ± 0.153	0.798 ± 0.046	0.866 ± 0.036	0.792 ± 0.018	0.998 ± 0.006	0.986 ± 0.012
generative	0.988 ± 0.004	0.776 ± 0.064	0.767 ± 0.084	0.787 ± 0.032	0.862 ± 0.023	0.758 ± 0.022	0.990 ± 0.026	0.982 ± 0.008
pertNBOTE	0.999 ± 0.001	0.755 ± 0.084	0.627 ± 0.077	0.818 ± 0.040	0.848 ± 0.035	0.767 ± 0.015	0.992 ± 0.010	0.981 ± 0.009
pertrand	0.999 ± 0.002	0.766 ± 0.074	0.620 ± 0.080	0.803 ± 0.032	0.850 ± 0.030	0.769 ± 0.021	0.994 ± 0.009	0.983 ± 0.011
randover	0.999 ± 0.001	0.788 ± 0.086	0.554 ± 0.158	0.799 ± 0.044	0.868 ± 0.039	0.794 ± 0.018	0.998 ± 0.006	0.988 ± 0.013
SMOTE	0.999 ± 0.001	0.813 ± 0.078	0.523 ± 0.169	0.810 ± 0.042	0.885 ± 0.038	0.814 ± 0.018	0.998 ± 0.007	0.987 ± 0.014
SMOTE2	0.999 ± 0.001	0.815 ± 0.091	0.501 ± 0.176	0.823 ± 0.045	0.901 ± 0.041	0.816 ± 0.021	1.000 ± 0.000	0.987 ± 0.014
Distant1	0.703 ± 0.044	0.129 ± 0.014	0.192 ± 0.037	0.183 ± 0.012	0.176 ± 0.026	0.281 ± 0.010	0.044 ± 0.003	0.272 ± 0.140
Distant2	0.239 ± 0.008	0.148 ± 0.019	0.217 ± 0.031	0.191 ± 0.015	0.149 ± 0.019	0.410 ± 0.012	0.595 ± 0.131	0.655 ± 0.168
Distant3	0.148 ± 0.001	0.051 ± 0.000	0.026 ± 0.000	0.090 ± 0.001	0.090 ± 0.003	0.089 ± 0.009	0.140 ± 0.102	0.188 ± 0.105
NearMiss1	0.875 ± 0.051	0.047 ± 0.002	0.163 ± 0.227	0.198 ± 0.025	0.204 ± 0.019	0.159 ± 0.024	0.042 ± 0.006	0.263 ± 0.308
NearMiss2	0.197 ± 0.003	0.090 ± 0.037	0.414 ± 0.052	0.221 ± 0.020	0.311 ± 0.031	0.598 ± 0.027	0.789 ± 0.101	0.814 ± 0.144
NearMiss3	0.935 ± 0.019	0.063 ± 0.014	0.027 ± 0.009	0.117 ± 0.010	0.162 ± 0.037	0.266 ± 0.072	0.781 ± 0.095	0.913 ± 0.060
randunder	0.979 ± 0.008	0.297 ± 0.040	0.388 ± 0.032	0.362 ± 0.026	0.431 ± 0.046	0.500 ± 0.020	0.963 ± 0.019	0.821 ± 0.093
SVM: Recall								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.930 ± 0.010	0.240 ± 0.051	0.143 ± 0.059	0.204 ± 0.026	0.335 ± 0.039	0.618 ± 0.030	0.735 ± 0.048	0.831 ± 0.057
GAIOTE	0.936 ± 0.010	0.355 ± 0.082	0.370 ± 0.110	0.387 ± 0.035	0.474 ± 0.040	0.738 ± 0.028	0.691 ± 0.052	0.762 ± 0.067
nonnGAIOTE	0.949 ± 0.013	0.390 ± 0.085	0.420 ± 0.104	0.424 ± 0.035	0.530 ± 0.037	0.758 ± 0.030	0.694 ± 0.054	0.818 ± 0.056
nonnSMOTE	0.907 ± 0.014	0.324 ± 0.068	0.260 ± 0.098	0.305 ± 0.027	0.402 ± 0.041	0.640 ± 0.027	0.615 ± 0.042	0.666 ± 0.065
NBOTE	0.924 ± 0.012	0.353 ± 0.069	0.300 ± 0.094	0.347 ± 0.036	0.445 ± 0.046	0.674 ± 0.028	0.653 ± 0.052	0.725 ± 0.069
generative	0.982 ± 0.005	0.348 ± 0.067	0.627 ± 0.072	0.403 ± 0.046	0.510 ± 0.045	0.732 ± 0.029	0.815 ± 0.041	0.908 ± 0.031
pertNBOTE	0.953 ± 0.011	0.352 ± 0.072	0.590 ± 0.063	0.428 ± 0.036	0.531 ± 0.035	0.701 ± 0.032	0.726 ± 0.047	0.864 ± 0.045
pertrand	0.955 ± 0.007	0.357 ± 0.086	0.583 ± 0.069	0.421 ± 0.036	0.526 ± 0.041	0.704 ± 0.030	0.732 ± 0.045	0.857 ± 0.051
randover	0.922 ± 0.012	0.359 ± 0.064	0.297 ± 0.099	0.346 ± 0.039	0.444 ± 0.045	0.672 ± 0.029	0.663 ± 0.051	0.707 ± 0.063
SMOTE	0.909 ± 0.015	0.333 ± 0.068	0.250 ± 0.085	0.307 ± 0.030	0.414 ± 0.040	0.640 ± 0.025	0.629 ± 0.042	0.656 ± 0.062
SMOTE2	0.907 ± 0.017	0.298 ± 0.058	0.223 ± 0.085	0.277 ± 0.025	0.384 ± 0.040	0.629 ± 0.031	0.612 ± 0.053	0.631 ± 0.057
Distant1	0.997 ± 0.003	0.931 ± 0.059	0.997 ± 0.011	0.926 ± 0.017	0.965 ± 0.007	0.963 ± 0.009	1.000 ± 0.000	0.984 ± 0.013
Distant2	0.995 ± 0.002	0.897 ± 0.061	0.997 ± 0.011	0.951 ± 0.014	0.978 ± 0.009	0.944 ± 0.012	0.956 ± 0.023	0.969 ± 0.027
Distant3	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.996 ± 0.004	1.000 ± 0.000	0.995 ± 0.003	0.990 ± 0.014	0.987 ± 0.013
NearMiss1	0.990 ± 0.004	0.821 ± 0.036	0.763 ± 0.087	0.852 ± 0.040	0.883 ± 0.053	0.949 ± 0.009	0.997 ± 0.006	0.972 ± 0.021
NearMiss2	1.000 ± 0.000	0.817 ± 0.045	0.960 ± 0.044	0.861 ± 0.037	0.815 ± 0.048	0.871 ± 0.024	0.926 ± 0.029	0.951 ± 0.030
NearMiss3	0.989 ± 0.004	0.766 ± 0.083	0.847 ± 0.065	0.957 ± 0.027	0.924 ± 0.053	0.872 ± 0.019	0.963 ± 0.014	0.941 ± 0.019
randunder	0.986 ± 0.003	0.790 ± 0.061	0.953 ± 0.036	0.796 ± 0.036	0.808 ± 0.045	0.918 ± 0.020	0.918 ± 0.025	0.961 ± 0.028
SVM: F-measure								
	classic	hitech	k1b	la1	la2	ohscal	reviews	sports
no_resampling	0.963 ± 0.006	0.374 ± 0.064	0.230 ± 0.087	0.326 ± 0.035	0.491 ± 0.043	0.718 ± 0.021	0.847 ± 0.032	0.898 ± 0.034
GAIOTE	0.966 ± 0.006	0.476 ± 0.082	0.444 ± 0.118	0.524 ± 0.036	0.611 ± 0.036	0.744 ± 0.010	0.814 ± 0.036	0.859 ± 0.045
nonnGAIOTE	0.973 ± 0.006	0.501 ± 0.078	0.487 ± 0.109	0.549 ± 0.025	0.652 ± 0.033	0.734 ± 0.008	0.816 ± 0.037	0.893 ± 0.036
nonnSMOTE	0.951 ± 0.008	0.458 ± 0.070	0.348 ± 0.123	0.443 ± 0.030	0.551 ± 0.041	0.715 ± 0.015	0.760 ± 0.031	0.794 ± 0.050
NBOTE	0.960 ± 0.007	0.483 ± 0.068	0.389 ± 0.116	0.483 ± 0.035	0.587 ± 0.045	0.728 ± 0.012	0.788 ± 0.036	0.834 ± 0.048
generative	0.985 ± 0.002	0.477 ± 0.062	0.687 ± 0.063	0.531 ± 0.039	0.640 ± 0.038	0.744 ± 0.010	0.893 ± 0.027	0.944 ± 0.018
pertNBOTE	0.975 ± 0.006	0.476 ± 0.068	0.606 ± 0.058	0.561 ± 0.030	0.653 ± 0.032	0.732 ± 0.014	0.838 ± 0.031	0.918 ± 0.027
pertrand	0.976 ± 0.004	0.481 ± 0.079	0.599 ± 0.064	0.551 ± 0.030	0.649 ± 0.037	0.734 ± 0.013	0.843 ± 0.029	0.915 ± 0.031
randover	0.959 ± 0.007	0.489 ± 0.064	0.386 ± 0.121	0.482 ± 0.037	0.586 ± 0.045	0.727 ± 0.013	0.796 ± 0.036	0.823 ± 0.047
SMOTE	0.952 ± 0.009	0.468 ± 0.068	0.338 ± 0.112	0.445 ± 0.033	0.563 ± 0.040	0.716 ± 0.014	0.771 ± 0.031	0.787 ± 0.048
SMOTE2	0.951 ± 0.009	0.433 ± 0.065	0.308 ± 0.114	0.414 ± 0.032	0.537 ± 0.041	0.710 ± 0.021	0.758 ± 0.041	0.769 ± 0.045
Distant1	0.824 ± 0.030	0.225 ± 0.021	0.320 ± 0.051	0.305 ± 0.016	0.297 ± 0.037	0.435 ± 0.012	0.084 ± 0.005	0.410 ± 0.148
Distant2	0.385 ± 0.010	0.254 ± 0.029	0.355 ± 0.042	0.318 ± 0.020	0.258 ± 0.029	0.571 ± 0.011	0.725 ± 0.097	0.768 ± 0.123
Distant3	0.257 ± 0.002	0.098 ± 0.001	0.051 ± 0.000	0.164 ± 0.002	0.164 ± 0.005	0.164 ± 0.015	0.233 ± 0.147	0.304 ± 0.138
NearMiss1	0.928 ± 0.030	0.089 ± 0.003	0.205 ± 0.235	0.321 ± 0.032	0.331 ± 0.023	0.271 ± 0.035	0.081 ± 0.011	0.335 ± 0.341
NearMiss2	0.330 ± 0.004	0.160 ± 0.055	0.577 ± 0.048	0.352 ± 0.023	0.449 ± 0.029	0.709 ± 0.017	0.849 ± 0.059	0.869 ± 0.088
NearMiss3	0.961 ± 0.009	0.116 ± 0.022	0.052 ± 0.016	0.208 ± 0.015	0.273 ± 0.050	0.402 ± 0.079	0.860 ± 0.061	0.926 ± 0.033
randunder	0.983 ± 0.003	0.430 ± 0.045	0.550 ± 0.029	0.497 ± 0.021	0.561 ± 0.039	0.647 ± 0.016	0.940 ± 0.016	0.881 ± 0.051

Table 13: Results for SVM with linear kernel

References

Batista, G, Prati, R., Monard, M. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data, SIGKDD Volume 6, Issue 1, 2004.

Barandela, R., Sanchez, J, Garcia, V, Rangel, E. Strategies for learning in class imbalance problems, Pattern Recognition 36, pp. 849-851, 2003.

Brank, J., Grobelnik, M., Milić-Frayling, N, Mladenić, D. Training text classifiers with SVM on very few positive examples; technical report, MSR-TR-2003-34, 2003.

Chawla, N.V., Bowyer, K.W., Hall, L.O. & Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-sampling TEchnique , Journal of Artificial Intelligence Research (JAIR), Volume 16, pp. 321-357, 2002.

Cover, T. M., Hart, P. E. Nearest neighbor pattern classification, IEEE Trans. on Information Theory 13, pp. 21-27, 1967.

Drummond, C. Holte, R. C4.5, Class Imbalance, and Cost Sensitivity: Why Under-Sampling beats Over-Sampling, ICML, 2003.

Japkowicz, N. Concept-Learning in the Presence of Between-Class and Within-Class Imbalances. Advances in Artifical Intelligence: Proceedings of the 14th Conference of the Canadian Society for Computational Studies of Intelligence, pp. 67-77, 2001.

Japkowicz, N. Class Imbalances: Are we Focusing on the Right Issue? ICML, 2003.

Japkowicz, N. and Stephen, S. The Class Imbalance Problem: A Systematic Study , Intelligent Data Analysis Journal, Volume 6, Number 5, November 2002.

Jo, T. and Japkowicz, N., Class Imbalances versus Small Disjuncts, SIGKDD Volume 6, Issue1, 2004.

Joachims,T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features, Proceedings of the European Conference on Machine Learning, Springer, 1998.

Joachims, T. Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

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

Ling, C, Li, C. Data Mining for Direct marketing: Problems and Solutions. KDD-98.

Maloof, M. Learning when data sets are imbalanced and when costs are unequal and unknown. ICML, 2003.

McCallum, A., Nigam, K. A Comparison of Event Models for Naive Bayes Text Classification, AAAI/ICML-98 Workshop on Learning for Text Categorization, pp. 41-48. Technical Report WS-98-05. AAAI Press. 1998.

Nickerson, A., Japkowicz, N. and Milios, E. Using Unsupervised Learning to Guide Resampling in Imbalanced Data Sets , in Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics, 2001.

Serrano, J., Castillo, M. del. A Multistrategy Approach for Digital Text Categorization from Imbalanced Documents, SIGKDD Volume 6, Issue 1, 2004.

Weiss, G, Provost, F. Learning When Training Data are Costly: The Effect of Class Distribution on Tree Induction, JAIR 19, pp. 315-354, 2003.

Wu, G., Chang, E. Class-Boundary Alignment for Imbalanced Dataset Learning, ICML, 2003.

Zhang, J., Mani, I. kNN Approach to Unbalanced Data Distributions: A Case Study involving Information Extraction, ICML, 2003.

Zheng, Z., Wu, X., Srihari, R. Optimally combining positive and negative features for text categorization, ICML, 2003.

Zheng, Z., Wu, X., Srihari, R. Feature Selection for Text Categorization on Imbalanced Data, SIGKDD Volume 6, Issue 1, 2004.

Zhong, S., Ghosh, J., A Comparative Study of Generative Models for Document Clustering, SDM Workshop on Clustering High Dimensional Data and Its Applications, May 2003.

VITA

Alexander Yun-chung Liu was born on January 9, 1980 in Atlanta, Georgia to Tung-lin and Shi Liu. The author graduated with a high school diploma from Clear Lake High School in May, 1998 and with a Bachelor of Science in Electrical Engineering from the University of Texas at Austin in December 2001. The author entered graduate studies at the University of Texas at Austin in August 2002. Throughout his education, the author has been supported by various generous donors, including a Texas Exes Award for Scholarship and Leadership, a Thrust Fellowship, and the Basdall Gardner Memorial Graduate MCD Fellowship in Engineering.

Permanent Address: 14206 Redbud Valley Trail
Houston, Texas 77062

This thesis was typed by the author.