Imbalance Learning Using Thresholding and Sample Repositioning

Hossein Ghaderi Zefrehi

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Eastern Mediterranean University
August 2023
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. Zeki Bayram
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. Hakan Altınçay
Supervisor

Examining Committee

1. Prof. Dr. A. Aydın Alatan
2. Prof. Dr. Hakan Altınçay
3. Prof. Dr. Gözde Bozdağı Akar
4. Prof. Dr. Ekrem Varoğlu
5. Assoc. Prof. Dr. Nazife Dimililer

ABSTRACT

The most frequently used approaches for imbalance learning are balancing by resampling, cost-sensitive learning and thresholding. In the balancing technique, the minority class is oversampled. Most of the algorithms used for this purpose are variants of the well-known algorithm named SMOTE, which is based on creating synthetic samples on the lines connecting selected minority instances. In cost-sensitive learning, the penalty of misclassifying a minority sample is set to be higher than that of a majority instance. In the thresholding approach, the decision threshold is adjusted to detect the minority class at the cost of increased misclassification of the majority instances.

In this thesis, the dependence of the optimal threshold on the performance metric is first studied. It is shown that the optimal thresholds for two widely used performance evaluation metrics, namely $F-score$ and $G-mean$ are different in most of the cases. In order to tackle the threshold estimation problem, building a threshold prediction model is defined as a meta-learning task. Novel features are suggested to quantify the imbalance characteristics of the datasets and the patterns among the prediction scores. The proposed threshold prediction model is built using these features extracted from external data. The model obtained is then employed to estimate the optimal thresholds for previously unseen datasets.

Repositioning of samples instead of balancing is also addressed. The classifiers are enforced to learn the decision region of the minority class by mainly repositioning the majority class samples. By repositioning, the regions in which minority instances exist are not outnumbered by the majority class samples. Hence, the classifier labels these

regions as the minority class. The minority samples are repositioned in small steps to avoid distorting the original distribution. The potential of the proposed repositioning scheme is also evaluated as a preprocessing algorithm for SMOTE.

Keywords: imbalance learning, repositioning, thresholding, balancing, binary classification, SMOTE

ÖZ

Dengesiz öğrenme için en sık kullanılan yaklaşımlar örnekleyerek dengeleme, maliyete duyarlı öğrenme ve eşiklemedir. Dengeleme tekniğinde, azınlık sınıfı aşırı örneklenir. Bu amaçla kullanılan algoritmaların çoğu, seçilen azınlık örneklerini birleştiren hatlar üzerinde sentetik örnekler oluşturmaya dayanan SMOTE adlı iyi bilinen algoritmanın varyantlarıdır. Maliyete duyarlı öğrenmede, bir azınlık örneğini yanlış sınıflandırmanın cezası, çoğunluk örneğininkinden daha yüksek olacak şekilde ayarlanmıştır. Eşikleme yaklaşımında, çoğunluk örneklerinin yanlış sınıflandırılması pahasına azınlık sınıfını tespit etmek için karar eşiği ayarlanır.

Bu tezde ilk olarak optimal eşiğin performans metriğine bağımlılığı incelenmiştir. Yaygın olarak kullanılan iki performans değerlendirme metriği, yani $F-score$ ve $G-mean$ için en uygun eşiklerin çoğu durumda farklı olduğu gösterilmiştir. Eşik tahmin probleminin üstesinden gelmek için eşik tahmin modeli oluşturmak, bir meta-öğrenme görevi olarak tanımlanmıştır. Veri setlerinin dengesizlik özelliklerini ve tahmin skorlarındaki örüntüleri ölçmek için yeni öznitelikler önerilmiştir. Önerilen eşik tahmin modeli, dış verilerden hesaplanan bu öznitelikleri kullanılarak oluşturulmuştur. Elde edilen model, daha önce görülmemiş veri kümeleri için en uygun eşikleri tahmin etmek için kullanılmıştır.

Dengeleme yerine örneklerin yeniden konumlandırılması da ele alınmaıştır. Sınıflandırıcılar öncelikle çoğunluk sınıfının örneklerini yeniden konumlandırarak azınlık sınıfının karar bölgelerini öğrenmeye zorlanır. Kaydırma neticesinde, azınlık örneklerinin bulunduğu bölgelerde, çoğunluk sınıfının örnek sayısı bakımından üstün olmaması sağlanır. Dolayısıyla, sınıflandırıcı bu bölgeleri azınlık sınıfı olarak

etiketler. Azınlık örnekleri, orijinal dağılımı bozmamak için küçük adımlarla yeniden konumlandırılmaktadır. Önerilen yeniden konumlandırma algoritmasının potansiyeli, SMOTE için bir ön işleme algoritması olarak da değerlendirilirmiştir.

Anahtar Kelimeler: dengesizlik öğrenme, yeniden konumlandırma, eşikleme, dengeleme, ikili sınıflandırma, SMOTE

Dedicated to

To my dearest wife, you are the anchor that steadies my soul and the joy that brightens my days. Your love and unwavering support have been the foundation of our beautiful family. To my precious daughters, you are the light of my life and the source of boundless pride. Your laughter and innocence fill my heart with happiness. And to my beloved parents, your guidance and love have shaped the person I am today. Your sacrifices and wisdom have been a guiding force throughout my journey. This thesis is dedicated to each of you, for you are the pillars of love that enrich my world with meaning and purpose. Together, you complete the very essence of my existence, and I am forever grateful for the cherished memories we share.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor Prof. Dr. Hakan Altınçay for the continuous support of my study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better supervisor and mentor for my study. Thank you professor!

I wish to express my gratitude to the members of the jury, namely Prof. Dr. Ekrem Varoğlu and Assoc. Prof. Dr. Nazife Dimililer, for their invaluable guidance and insightful corrections. Their assistance played a crucial role in unifying and refining this thesis, and significant enhancements have been achieved, all thanks to your constructive input.

I would also like to express my heartfelt appreciation to all the Professors, employees, and assistants in the computer engineering department. Your continuous support and encouragement were immensely valuable throughout my studies. Thank you for your assistance, which has been invaluable to me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AUC	Area Under ROC Curve
CD	Critical Difference
FN	False Negative
FP	False Positive
LR	Logistic Regression
MaMiPot	MAjority and MInority rePOsitioning Technique
NB	Naive Bayes
RF	random forest
SMOTE	Synthetic Minority Oversampling TEchnique
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

Chapter 1

INTRODUCTION

1.1 Background

A challenge for many single and multi-label classification tasks is class imbalance. In single-label classification, some classes may have significantly fewer training samples than the others due to the nature of the classification problem [1, 2]. For instance, in many real-world binary pattern classification tasks such as detection of a particular disease like cancer [3], software defect prediction [4], spam filtering [5], fraud detection [6, 7], churn prediction [8] and intrusion detection [9], the target (or, positive) class is generally the minority, including far less samples compared to the majority (or, negative) class. On the other hand, solving multi-label classification tasks such as text categorization and emotion classification in musical recordings leads to an even more significant challenge [10–12]. A widely-used approach to address multi-label classification is problem transformation [13–15]. In this technique, the problem is formulated as multiple single-label binary classification tasks where each task is defined for a particular label. More specifically, the positive class includes the samples having the target label whereas the rest of the samples form the negative class. This approach further increases the inherent degree of imbalance. When other distribution-dependent factors such as the availability of rare instances representing infrequent sub-concepts and overlaps between minority and majority classes are combined with the bias on the majority class due to imbalance, the classification problem becomes a highly complex task. [16]. Most of the

conventionally used learners achieve very poor performance on the minority class since they compute decision regions by considering the number of classification errors on the whole training data [17]. In some cases, due to large class imbalance and/or having only a few positive samples, the learner almost always assigns the labels as the majority class, leading to a very high but spurious accuracy value.

In order to improve the performance on the minority class without severely deteriorating that of the majority, various techniques are proposed [18–21]. The most widely used approach is *balancing* the training set by resampling. In this technique, either the number of minority samples is increased or the number of majority samples is decreased [22]. After balancing by resampling, some parts of the feature space where the minority samples were outnumbered by the negatives will be learned as positive decision regions. For instance, in random oversampling, the minority class is oversampled by creating several copies of the available samples [23]. As another approach of oversampling, generating artificial minority samples is addressed and numerous techniques have been proposed, most of which are variants of a well-known algorithm named Synthetic Minority Oversampling TEchnique (SMOTE) [24, 25]. On the other hand, a random subset of the majority samples is selected in random undersampling [26]. Alternatively, Wilson's Edited Nearest Neighbor rule discards the samples for which more than one of the three nearest neighbors are from the other class [27]. In the condensed nearest neighbor classifier, it is aimed at finding a consistent subset of samples that correctly classifies all training samples using nearest neighbors [28]. Radial-based undersampling is a recent approach that is based on the mutual class potential of the majority samples [22]. In Tomek links approach, it is aimed to identify samples from different classes that are closer to each other than the samples from their own class [29]. The majority samples forming Tomek links are

then removed for undersampling the majority class.

Cost-sensitive learning is another technique employed in imbalance learning [30, 31]. In this approach, the bias on the majority class is reduced by defining a higher misclassification cost for the minority class than that of the majority. For instance, the penalty factor is defined to be higher for the minority class in training SVM [32, 33]. Similarly, a weighted cross-entropy loss is employed for XGBoost to consider misclassified positive samples as larger losses compared to the misclassified negatives [34]. Applying instance weighting to develop cost-sensitive decision trees [35], utilizing different misclassification costs in naive Bayes classification [36], correcting the network outputs during training of neural networks [37] and hybrid approaches of resampling and cost-sensitive learning are also studied [30].

Thresholding (or, threshold-moving, post-scaling) is a post-training solution to class imbalance [38–43]. Thresholding is extensively studied for multi-label classification. However, in single-label tasks, it has attracted less interest when compared to balancing and cost-sensitive learning. This approach mainly aims at tackling poorly calibrated probability scores due to class imbalance. It is applied in two different ways. In the former approach, the best-fitting decision threshold that is expected to be different from the default, i.e. 0.5 is computed using the training data. In the latter, the threshold is calculated by applying cross-validation on the training data [40]. However, the positive class is rare in some cases and, computing the best-fitting threshold is highly challenging due to the risk of overfitting [39].

Hybrid (or ensembling) techniques form the most widely used group since they benefit from the diversity among the members constructed using different techniques

such as balancing or cost-sensitive learning. For instance, instead of discarding most of the potentially useful majority samples as practiced in random undersampling [44], RUSboost trains each member using a different subset of the majority samples, effectively utilizing all available data [45]. In BalanceCascade, the majority instances that are correctly classified by previous learners are discarded before training the following members [46].

1.2 Motivation

Imbalance learning using balancing has been extensively studied in the last two decades and more than a hundred variants of SMOTE are proposed [25]. Balancing aims to enforce the learner to label the decision regions where the minority samples appear as minority regions to improve the true positive rate. However, generating synthetic samples may lead to some drawbacks. For instance, the true boundary may change if an outlier is employed for generating synthetic samples. Similarly, irrelevant positive regions may be generated. The assumptions made by some classifiers about the original class distribution such as the normality of feature values may be violated due to changes in distributions. On the other hand, thresholding does not suffer from these drawbacks since the models are computed using original instances.

1.3 Contributions

In this thesis, a meta-learning-based threshold prediction technique is proposed. Seventeen meta-features are defined to construct a meta-space for the meta-learner. These features mainly represent the distribution of the positive and negative samples in the score space. Using this representation, threshold prediction is defined as a mapping that does not heavily depend on the task under concern, such as the number of input features and the number of training samples. The meta-learner predicts a threshold to be employed during testing by using the meta-space representation of the

scores obtained by testing the classifier under concern with the training data.

We also propose repositioning the negative samples toward their centroid to decrease the number of majority samples in the subspace where the minority samples are present. The positive samples are also relocated, but only slightly to deal with noisy or outlier examples. By altering the positions of the minority samples only slightly, it is aimed not to change the position of positive borderline samples and prevent the boundary from being distorted. Moreover, irrelevant positive regions are not generated since the minority instances are slightly modified.

1.4 Outline

Chapter 2 presents a literature review. In Chapter 3, a meta-learning-based thresholding approach for predicting the threshold for imbalanced datasets is introduced. Imbalance learning using sample repositioning is presented in Chapter 4. Experimental settings and the results obtained are given in Chapter 5. Finally, in Chapter 6 conclusions and suggestions for future studies are presented.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

The solutions for binary imbalanced classification tasks are generally easier to formulate when compared to multi-class setting. For instance, samples can be categorized by taking into account their positions with respect to the decision boundary, such as boundary sample, safe or outlier [16]. Since there are two classes, each class is either minority or majority. However, multi-class problems are more challenging, mainly due to having more complex relations between the classes than the binary case [47, 48]. Moreover, learning the decision boundaries is expected to be more difficult in the multi-class case.

Plenty of methods are developed for binary imbalanced classification. These techniques are generally categorized as *algorithm level*, *data level* and *hybrid* (or, ensemble) approaches [18, 21, 49, 50]. In algorithm level methods, misclassification of positives is assigned a larger cost compared to negatives [51–54]. In particular, a larger cost is assigned to a false negative. Similarly, thresholding methods aim to increase the number of true positives by estimating a better decision threshold than the default [55]. Data level techniques balance the classes by applying resampling [24]. On the other hand, hybrid methods target utilizing different techniques together with ensemble methods such as bagging and boosting [19, 49, 50, 56].

2.2 Algorithm Level Approaches

In algorithm-level modifications of existing classifiers, different costs can be assigned to different types of errors. For instance, in cost-sensitive decision trees, the cost of misclassifying a minority class sample is set to be higher than a majority sample [57]. The penalty of misclassifying minority class samples by a support vector machine classifier is set to be higher than that of the majority class [58]. XGBoost uses a weighted cross-entropy loss to treat misclassified positive samples as bigger losses than misclassified negative ones [34].

Thresholding is another algorithm level approach for dealing with imbalance. However, in cases of high imbalance ratios, estimating the best-fitting threshold is a challenging task [40]. It has been reported in Ref. [59] that the threshold corresponding to a balanced sensitivity and specificity is between the proportions of minority and majority classes, which is a large interval when the imbalance ratio is large. Because of these reasons, the proportion of the minority class and the difference between the number of positive and negative training samples are generally used in defining alternative decision thresholds [60]. For instance, the midway between the probability of the minority class and 0.5 is proposed as a threshold-moving solution [43]. Calibrating the output scores is an analogous technique to thresholding [61]. For instance, the output of the convolutional neural networks for each class is divided by the a priori probability of the class to obtain accurate class probabilities [62]. In binary imbalance classification problems, this implies setting the decision threshold as the a priori probability of the minority class. Experimental results on three image classification datasets have proven the effectiveness of thresholding by compensating prior class probabilities [62]. Saerens *et al.* suggests a transformation to adjust the outputs of classifiers to improve

classification performance in the case of having different a priori probabilities in the training and test data [63]. Threshold-moving is also implemented by scaling the classifier outputs using misclassification costs and it is argued to be an "excellent" solution to imbalance learning in binary classification problems [61].

In order to estimate the best-fitting threshold for the test data, the classification scores obtained using training data can be employed. In this approach, the decision threshold is computed as the value that maximizes the performance metric by testing the classifier using the training data [55]. As an equivalent approach to calibrating the output scores by a priori probabilities, the proportion of the minority samples can be defined as the threshold [62]. More specifically, let p and $(1-p)$ respectively denote the probability scores of a test sample for the minority and majority classes. Assume that the proportion of minority and majority samples are P_{min} and P_{maj} respectively, where $P_{min}+P_{maj}=1$. The decision will be on minority sample if $\frac{p}{P_{min}} > \frac{(1-p)}{P_{maj}}$, or equivalently $p > P_{min}$ [43]. The midway between the proportion of the minority class and 0.5 is a recently proposed approach to set the threshold [43]. The main motivation is the fact 0.5 is shown to be the upper bound for the threshold that maximizes $F-score$ [42]. Lipton *et al.* have shown that, when the output scores are well calibrated, the optimal threshold that maximizes $F-score$ is its half [42]. This corresponds to selecting the threshold as half of the maximum $F-score$ that can be achieved by testing the classifier with its training data for all candidate thresholds. SVM-THR sets the threshold of SVM scores as $(P-N)/(P+N+2a)$ where P and N respectively denote the numbers of positive and negative samples [60]. The default value of a is one but it can be estimated from the training data to adjust the threshold. In SVM-OTHR, the optimal threshold is computed by an exhaustive search on the misclassified minority samples [41]. In particular, it is computed by using the

distance between each misclassified positive sample and its nearest neighbor from the negative class.

An alternative approach to compute the best-fitting threshold is using a threshold predictor that is referred to as a MetaLabeler (or, meta-learner) [64]. To the best of our knowledge, this technique has been used only in multi-label tasks where the decision threshold is instance-based [65]. The input space of the meta-learner may be defined as the original input space, ranks of different labels, or scores of the labels [64]. During testing, the sample is assigned all the labels with scores above the estimated threshold [66]. Instead of predicting the threshold, a meta-learner can also be trained to compute the number of labels of a given instance [64]. In that case, the top-ranked labels are selected according to the predicted numbers.

2.3 Data Level Approaches

Balancing using resampling is a widely-used approach to tackle the imbalance problem. In the random undersampling approach, a subset of the majority instances is randomly selected. On the other hand, random oversampling duplicates the minority samples for this purpose. In Synthetic Minority Over-sampling Technique [67] (SMOTE), synthetic minority samples are generated. In this approach, new instances are generated on lines connecting existing minority samples. Particularly, given a minority point named a *seed*, a sample from its k-nearest minority samples is randomly selected. Then, a synthetic sample is generated on the line joining these two points. This process is repeated until the desired number of synthetic samples is obtained.

SMOTE has been criticized to have several shortcomings [68], which can be itemized as follows: (1) The samples in dense clusters will have neighbors that are likely to be

from the same clusters. In such cases, the synthetic samples will be near replicas of the seed sample lying in dense clusters. (2) The relative positions and distances of the minority samples are not considered in k-nearest-based neighbor selection. Because of this, the selected samples may be in incorrect regions. For instance, the seed sample may be noisy or an outlier and hence located in a region that is far away from the other minority samples. In such a case, the synthetic samples may also be away from the other minority examples. (3) The nearest neighbor of a seed may be in a different cluster. In this type of case, the synthetic sample may be located in the space of the majority class. (4) Due to the linear interpolation-based sample generation, the true distribution of the minority samples will be distorted [69, 70]. If a classifier takes into account the variance of the minority samples, incorrect variance estimates will lead to additional challenges for the classifiers during testing [71]. Moreover, the training samples will not be independent after balancing, violating the independence assumption, if employed by the classifier [71]. (5) Using an a priori fixed k value is another weakness since variations of the density of samples in different regions is ignored. (6) All samples are not equally important from the oversampling point of view. For instance, some samples such as those located close to the decision boundary are harder-to-classify but SMOTE does not differentiate between the samples [16, 68].

More than a hundred variants of SMOTE have been proposed after it was published in year 2003 [24, 25]. These methods mainly differ in the approaches utilized in selecting and/or weighting the minority seed samples, the methods for generating new samples to replace linear interpolation, and the elimination of existing noisy samples and incorrect synthetic samples that are located in the majority class region.

In SMOTEFUNA [72], the furthest sample from the seed is selected and the synthetic

sample is generated within the cuboidal space, allowing to produce diverse synthetic instances. This technique is argued to have several advantages. For instance, the synthetic samples are not expected to be close to their seeds in the cases when the seed is from a dense cluster. Hence, the shortcoming mentioned in item (1) is improved. Moreover, since only the furthest instance is considered, setting the value of the tuning parameter k is not needed.

In order to avoid employing noisy instances as a seed that is mentioned in item (2), DBSMOTE utilizes a clustering algorithm [73]. The samples that do not lie in a cluster are labeled as noisy instances and they are not utilized in oversampling. In MSMOTE, if all k neighbors of a minority sample belong to the majority class, it is labeled as a noisy instance and ignored [74]. MWMOTE [68] checks the neighbors of all minority samples. The samples which do not have any minority example in their neighbor set are discarded. Hence, such samples will not be considered as either seed or neighbor during linear interpolation. In Safe-Level SMOTE, the safety of the two samples selected for linear interpolation is computed by checking their neighbors [75]. The synthetic sample is generated closer to the instance that is more safe.

As a solution to the weakness mentioned in item (3), Hu et al. [76] proposed to use a classifier to evaluate the validity of each synthetic sample. A classifier is initially trained using the training samples. Two randomly selected minority samples are employed to generate a new synthetic sample by linear interpolation. This sample is accepted if the confidence score of the classifier is within a predetermined confidence interval. In GSMOTE, a safe area is defined for each minority sample [77]. The safe area represents the feature space where the generated samples are not noisy. In

SMOTEFUNA [72] algorithm, the distances of each synthetic sample from the nearest minority and nearest majority sample are utilized to evaluate its validity. In particular, if a synthetic sample is closer to the negative nearest neighbor, it is discarded. In RBO, the synthetic samples are generated by computing Gaussian radial basis function-based distributions of samples in both positive and negative classes [78]. The difficult regions are identified by using these potential surfaces and oversampled. ADPCHFO proposed by Tao *et al.* applies oversampling by employing samples within the same clusters [79]. Density peak clustering is applied to cluster the minority samples. Using clusters allows the generation of samples within valid regions. In order to avoid overfitting due to duplicate samples, a heuristic filter is applied to eliminate overlapping instances.

GSMOTE [77] addresses the shortcoming mentioned in item (4) by using geometric regions rather than linear interpolation for generating synthetic instances. For each minority sample, a safe radius is computed. Based on this value, sample generation takes place in a truncated hyper-spheroid. Xie *et al.* proposed the use of Gaussian distribution around the seed samples for generating synthetic instances [70]. For a given seed, a random direction that originates from the seed is initially selected. The new sample is generated in that direction where the distance is computed using a Gaussian distribution whose parameters are computed from the training data. In ROSE [80], synthetic instances are sampled from a kernel density estimate that is centered at the selected samples. For both GSMOTE and ROSE, setting the value of the tuning parameter k is not needed. Hence, the shortcoming mentioned in item (5) is avoided. SWIM employs the density of the well-represented majority class for generating new samples [81]. In particular, the synthetic samples have approximately the same density as the seed.

In order to address the shortcoming mentioned in (6), ADASYN selects the seeds by considering the weights computed for each minority samples [82]. The weight is proportional to the number of neighbors from the majority class. Hence, the learner is expected to focus on the hard-to-classify samples. In order to put more emphasis on hard-to-classify samples, SDSMOTE identifies the borderline samples by defining the degree of support on the minority samples [83]. In MWMOTE, the hard-to-learn samples are selected by considering the distance of each minority sample to its closest majority instance [68].

Many other variants of SMOTE have also been proposed. For instance, SMOTE-IPF addresses the shortcoming in item (2) by removing both the noisy samples in the original data and noisy synthetic samples [84]. TRIM-SMOTE proposes preprocessing the minority instances to generate multiple seed sets to be utilized for generating synthetic instances [85]. In Polynom-fit-SMOTE method, synthetic instances are generated using polynomial fitting [86]. MCT generates synthetic samples using cloning where the instances on the borderline are cloned less than those that are internal to the minority class region [87]. SMOTE-D considers the distances between each seed and its k-nearest neighbors to calculate the number of synthetic instances between each sample pair [88]. Cluster-SMOTE aims at generating artificial samples within the major clusters of the minority class [89]. CURE-SMOTE is another clustering-based approach [90]. The minority samples are clustered to identify small clusters. These are assumed to be noisy and removed. MDO generates synthetic samples in dense regions of the minority class and hence reduces the risk of generating instances in majority class space [91]. ANS [92] automatically determines the number of neighbors and adapts it for different regions of minority samples. In this algorithm, the minority instances that have no neighbors

around themselves are labeled as outcasts and they are not utilized in the sample creation process. In kmeans-SMOTE, the training data is clustered in an unsupervised way [93]. The clusters including a small proportion of minority samples are not selected for oversampling. Moreover, dense minority clusters are oversampled more than sparse clusters. In Borderline-SMOTE, a subset of minority samples is considered as seed samples. For instance, if all k-nearest neighbors of a minority sample are from the majority class, it is assumed to be noise and not considered during oversampling. A sample in the remaining set is considered as a seed only if most of its nearest samples are from the majority class. This criterion is expected to be mainly satisfied by the borderline samples and hence the learner will more accurately learn the decision boundary. A variant of Borderline-SMOTE which takes into account the neighboring majority samples as well in generating synthetic samples is also proposed [94]. Data cleaning is an alternative technique used for post-processing the balanced datasets to correct the distortions of the class clusters. For instance, in SMOTE-ENN, any instance that is not correctly labeled by employing three nearest neighbors is deleted [95]. Similarly, SMOTE-TomekLinks discards the instances that form Tomek links [95]. CCR is another data-cleaning-based approach where, before oversampling, the neighborhood of each minority sample is cleaned by removing majority class instances [96].

Balancing is also addressed by analyzing the types of samples in the minority class. For instance, the minority samples are categorized as safe, rare, borderline, and outlier. Te weights of a one-class classifier are adjusted based on the types of samples [97]. Sampling by analyzing the neighborhood of the positive samples is also proposed [98]. A bagging-based ensemble of classifiers is constructed where each bootstrap is formed by mainly including hard-to-learn samples. In particular, the probability of drawing an

instance depends on its difficulty of classification which is quantified using the number of neighbors from the majority class.

2.4 Hybrid Approaches

Classifier ensembles which are based on employing balancing or misclassification costs in building the members are also widely used in imbalance learning since they allow benefiting from the diversity achieved by training the members under different settings [19, 99–102]. Both balancing and cost-sensitive learning techniques are criticized for producing miscalibrated probability scores [7, 43, 103]. When balancing is used, miscalibration occurs due to training the classifiers with class proportions different from the test set. It is generally argued that the probability estimate of the samples in the minority class is not generally improved by correcting the imbalance [104]. Because of miscalibrated probabilities, the default decision threshold is not generally the best-fitting, even after correcting the imbalance [40, 62]. Similarly, computing the misclassification costs is challenging in cost-sensitive learning [100]. On the other hand, hybrid approaches built using multiple members, each employing a different approach to tackle imbalance, are strong candidates to achieve better performance scores when compared to algorithm and data level techniques.

Chapter 3

THRESHOLD PREDICTION USING A META-LEARNING APPROACH

3.1 Introduction

A major challenge for thresholding-based imbalance learning is susceptibility to overfitting, especially in cases when the minority classes are rare, including only a few samples in some cases. Moreover, the best-fitting threshold is generally metric-dependent. Because of this, the target threshold employed in training the meta-learner must be selected by considering the performance metric. On the other hand, a natural consequence of metric dependency is that threshold prediction errors according to one metric may lead to improved performance scores for another. In order to deal with the challenges mentioned above, employing a meta-learner trained on external datasets for single-label threshold prediction is proposed. There are three main contributions in this approach. The first contribution is developing a meta-space representation of prediction scores. Seventeen meta-features are defined to construct a meta-space for the meta-learner. These features mainly represent the distribution of the positive and negative samples in the score space. Using this representation, threshold prediction is defined as a mapping that does not heavily depend on the task under concern, such as the number of input features and the number of training samples. The second contribution is utilizing external datasets in threshold prediction. In particular, a meta-learner operating on the defined meta-space that is trained using a diverse set of external datasets is employed to predict the best-fitting thresholds for

unseen meta-space representations. In fact, we were initially inspired by a previously studied approach for feature generation based on utilizing diverse data from a repository of datasets [105]. In that study, a ranking classifier is built from an external group of datasets to select a good set of features for unseen datasets. In the current study, the meta-learner is trained offline using external data for threshold prediction, as it was done in Ref. [105] for feature generation. The meta-learner predicts a threshold to be employed during testing by using the meta-space representation of the scores obtained by testing the classifier under concern with the training data.

For a deeper understanding of the way threshold prediction errors affect different metrics in the case of class imbalance, the relative values of best-fitting thresholds for different metrics should be investigated. As the third contribution, the relation between the optimal thresholds for $F-score$ and $G-mean$ that is based on the numbers of positive and negative training instances is derived. In fact, it is highly crucial to investigate the circumstances when two metrics have different optimal thresholds. In practice, this will aid in deciding whether to follow the general procedure and evaluate the imbalance learners in terms of multiple metrics using a common threshold or, to tune the threshold separately for each evaluation metric.

3.2 Performance Metrics

Accuracy is not an appropriate performance metric for imbalance learning. For instance, if the majority class consists of 99% of the dataset, a classifier that is not able to detect any positive sample will still be reported as highly accurate [100]. $F-score$, $G-mean$, and AUC are the most widely used metrics in imbalance learning. Let P and N denote the total numbers of positive and negative instances, respectively. $F-score$ is defined in terms of precision and recall as

$$Precision = \frac{TP}{TP+FP} \tag{3.1}$$

$$Recall = \frac{TP}{TP+FN} = \frac{TP}{P} \tag{3.2}$$

$$F-score = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP+FP+FN} \tag{3.3}$$

where TP, FP, and FN denote true positives, false positives, and false negatives, respectively. $G-mean$ is defined in terms of sensitivity (or, recall) and specificity as

$$Sensitivity = \frac{TP}{P} \tag{3.4}$$

$$Specificity = \frac{TN}{TN+FP} = \frac{TN}{N} \tag{3.5}$$

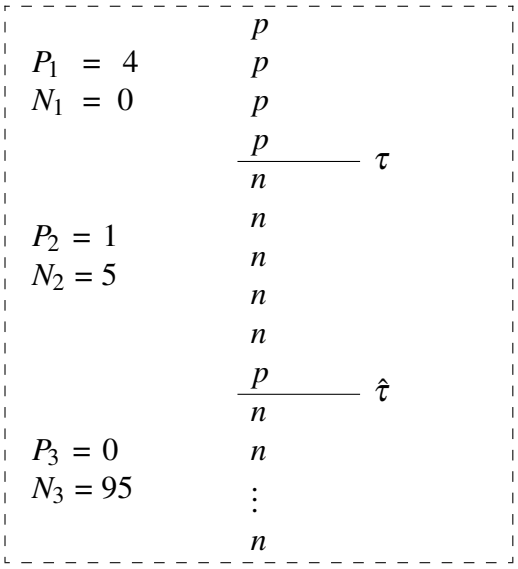
$$G-mean = \sqrt{Sensitivity \times Specificity} = \sqrt{\frac{TP \times TN}{PN}} \tag{3.6}$$

where TN denotes true negatives.

AUC is defined as the area under Receiver Operating Characteristics (ROC) curve which represents the relation between the true positive rate (TP/P) and the false positive rate (FP/N). AUC is independent of the threshold and widely used in imbalance classification.

3.3 Threshold-Moving for Different Performance Metrics

For evaluating classifiers in imbalance learning, measures based on combining two metrics, namely $F-score$ that considers both precision and recall and, $G-mean$ employing the product of sensitivity and specificity are used [99]. The effect of miscalibrated probability scores for different metrics may not be the same. Therefore, in defining the optimal threshold, the metric utilized for performance evaluation must also be considered. In order to gain a deeper understanding of the characteristics of the optimal thresholds for different metrics and investigate the circumstances when two metrics have different optimal thresholds, an analytical evaluation is performed. The best-fitting thresholds for $F-score$ and $G-mean$ are not necessarily the same. In



Metric	τ	$\hat{\tau}$
Precision	1	$1/2$
Recall	$4/5$	1
Sensitivity	$4/5$	1
Specificity	1	$95/100$
$F-score$	**0.8889**	0.6667
$G-mean$	0.8944	**0.9747**

Figure 3.1: An exemplar ranking of samples, performance scores, and optimal thresholds for $F-score$ and $G-mean$.

order to explore the relation between these thresholds, assume that the tested samples are sorted in descending order, using the scores generated by the classifier. An exemplar ranking of samples and two candidate thresholds, τ and $\hat{\tau}$, are illustrated in Figure 3.1. Let P_1 and N_1 respectively denote the number of positive and negative samples with scores greater than τ. P_2 and N_2 respectively denote the number of positive and negative samples with scores less than τ and greater than $\hat{\tau}$. Similarly, let P_3 and N_3 represent the number of positive and negative samples whose scores are less than $\hat{\tau}$.

Assume that the thresholds are computed as the midway between two consecutive scores. For instance, τ is the average of the scores assigned to the positive sample in the fourth rank and the negative sample in the fifth rank. In the following context, we will refer to the thresholds between a positive sample and its following negative sample as being located at a p-to-n transition. It can be seen that $\hat{\tau}$ is also at a p-to-n transition. The $F-score$ and $G-mean$ values at τ and $\hat{\tau}$ are also presented in the figure. The characteristics of the optimal threshold for $F-score$ are recently addressed

in [55] by deriving the expressions for $F-score$ at the scores of positive samples. In particular, when the samples are ranked according to their probability scores, it is shown that the optimal threshold is provided by the score of a positive sample. They have also proven that, for two consecutive positives, the $F-score$ that is obtained by thresholding at the score of the lower-ranked positive sample is higher than that of the higher-ranked sample. In fact, these observations can be better stated as follows: The optimal threshold for $F-score$ should be placed at a p-to-n transition. In Proposition 1 given below, this fact is proven in a more straightforward way in terms of recall and precision by using contradiction. Additionally, it is shown that the optimal threshold for $G-mean$ is similarly at a p-to-n transition by employing sensitivity and specificity.

Proposition 1: The optimal thresholds for both $F-score$ and $G-mean$ are at p-to-n transitions.

Proof: The proof can be done by contradiction. Consider a particular threshold τ is optimal. If τ is at a n-to-p or p-to-p transition, moving the threshold below the following p (i.e. decreasing) will increase the number of true positives and hence recall. The precision will not decrease (i.e. either increases or remains the same if equal to 1), leading to an increased $F-score$. Similarly, sensitivity will increase where specificity will remain the same. Hence, $G-mean$ will also increase. We conclude that τ can not be the optimal threshold.

If τ is at an n-to-n transition, increasing the threshold above the higher-ranked n will decrease the number of false positives and hence improve precision where the recall remains the same. Hence, $F-score$ will be increased. Similarly, sensitivity will remain the same whereas specificity will increase. Hence, $G-mean$ will also increase and thus

τ is not the optimal threshold. $\square$

The proof of Proposition 1 clearly shows that, if τ is at a p-to-n transition, it is a candidate for optimal threshold. For instance, the threshold below the lower-ranked n (i.e. decreasing) will correspond to an n-to-n or n-to-p transition that is not a candidate to be optimal. Similarly, the threshold above the higher-ranked p will correspond to a n-to-p or p-to-p transition that is not an optimal candidate. For the example given in Figure 3.1, since there are only two p-to-n transitions, we conclude using Theorem 1 that the optimal threshold for $F-score$ is τ whereas, $\hat{\tau}$ is the optimal threshold for $G-mean$. This example also shows that the optimal thresholds for the two metrics are not necessarily the same. A relation between these thresholds is presented and proven in Theorem 1.

Theorem 1: Let $S_\tau = \{\tau, \hat{\tau}\}$ denote a set of two p-to-n transitions where $\hat{\tau} < \tau$ and, assume that $\tau_F \in \{\tau, \hat{\tau}\}$ and $\tau_G \in \{\tau, \hat{\tau}\}$ respectively denote the best thresholds in S_τ for $F-score$ and $G-mean$. If $N_3 > (P + N_1)$, then $\tau_G \leq \tau_F$.

Proof: As given in Figure 3.1, assume P_1 positive and N_1 negative samples above τ, P_3 positive and N_3 negative samples below $\hat{\tau}$ and, P_2 positive and N_2 negative samples between τ and $\hat{\tau}$. Then, the metric values can be calculated for τ as:

$$
\begin{aligned}
Precision[\tau] &= \frac{P_1}{P_1+N_1} \\
Recall[\tau] &= \frac{P_1}{P} \\
Sensitivity[\tau] &= \frac{P_1}{P} \\
Specificity[\tau] &= \frac{N-N_1}{N} \\
F-score[\tau] &= \frac{\frac{2P_1P_1}{P(P_1+N_1)}}{\frac{P_1}{(P_1+N_1)}+\frac{P_1}{P}} = \frac{2P_1}{P+P_1+N_1} \\
G-mean[\tau] &= \sqrt{\frac{P_1(N-N_1)}{P\times N}}
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
Precision[\hat{\tau}] &= \frac{P_1+P_2}{P_1+P_2+N_1+N_2} \\
Recall[\hat{\tau}] &= \frac{P_1+P_2}{P} \\
Sensitivity[\hat{\tau}] &= \frac{P_1+P_2}{P} \\
Specificity[\hat{\tau}] &= \frac{N-N_1-N_2}{N} \\
F-score[\hat{\tau}] &= \frac{2(P_1+P_2)}{P+P_1+P_2+N_1+N_2} \\
G-mean[\hat{\tau}] &= \sqrt{\frac{(P_1+P_2)(N-N_1-N_2)}{P\times N}}
\end{aligned}
$$

The expression $F-score[\hat{\tau}] > F-score[\tau]$ implies

$$
\frac{2(P_1+P_2)}{P+P_1+P_2+N_1+N_2} > \frac{2P_1}{(P+P_1+N_1)}.
$$

After some manipulations as given below,

$$
(P_1+P_2)(P+P_1+N_1) > P_1(P+P_1+P_2+N_1+N_2)
$$

$$
\cancel{P_1P} + \cancel{P_1^2} + \cancel{P_1N_1} + P_2P + \cancel{P_2P_1} + P_2N_1 > \cancel{P_1P} + \cancel{P_1^2} + \cancel{P_1P_2} + \cancel{P_1N_1} + P_1N_2
$$

we obtain

$$
\frac{P_2}{N_2} > \frac{P_1}{P+N_1}.
$$

Similarly, $G{-}mean[\hat{\tau}] > G{-}mean[\tau]$ implies

$$\frac{(P_1+P_2)(N-N_1-N_2)}{P\times N} > \frac{P_1(N-N_1)}{P\times N}.$$

Note that N denotes the total number of negative samples, i.e. $N = N_1+N_2+N_3$. Hence $(N-N_1-N_2) = N_3$ and $(N-N_1) = (N_2+N_3)$ After some manipulations on the expression given above, we obtain

$$\begin{aligned}(P_1+P_2)(N-N_1-N_2) &> P_1(N-N_1)\\ (P_1+P_2)N_3 &> P_1(N_2+N_3)\\ \cancel{P_1N_3}+P_2N_3 &> P_1N_2+\cancel{P_1N_3}\\ \frac{P_2}{N_2} &> \frac{P_1}{N_3}.\end{aligned}$$

When $N_3 > (P+N_1)$,

$$F{-}score[\hat{\tau}] > F\_score[\tau] \;\Rightarrow\; G{-}mean[\hat{\tau}] > G{-}mean[\tau].$$

and,

$$G{-}mean[\hat{\tau}] > G{-}mean[\tau] \;\not\Rightarrow\; F{-}score[\hat{\tau}] > F{-}score[\tau].$$

Hence, it can be concluded that, $N_3 > (P+N_1) \Rightarrow \tau_G \leq \tau_F$.

In imbalanced classification problems, $N \gg P$. Since we expect the lower-ranked samples to be mainly negatives, it can be argued that the expression $N_3 > (P+N_1)$ is generally true in imbalanced classification. Theorem 1 clearly shows that threshold selection should take into account the performance metric. In other words, the performance of different imbalance learning techniques must be evaluated by using metric-dependent thresholds. In this thesis, threshold estimation for $F{-}score$ is addressed.

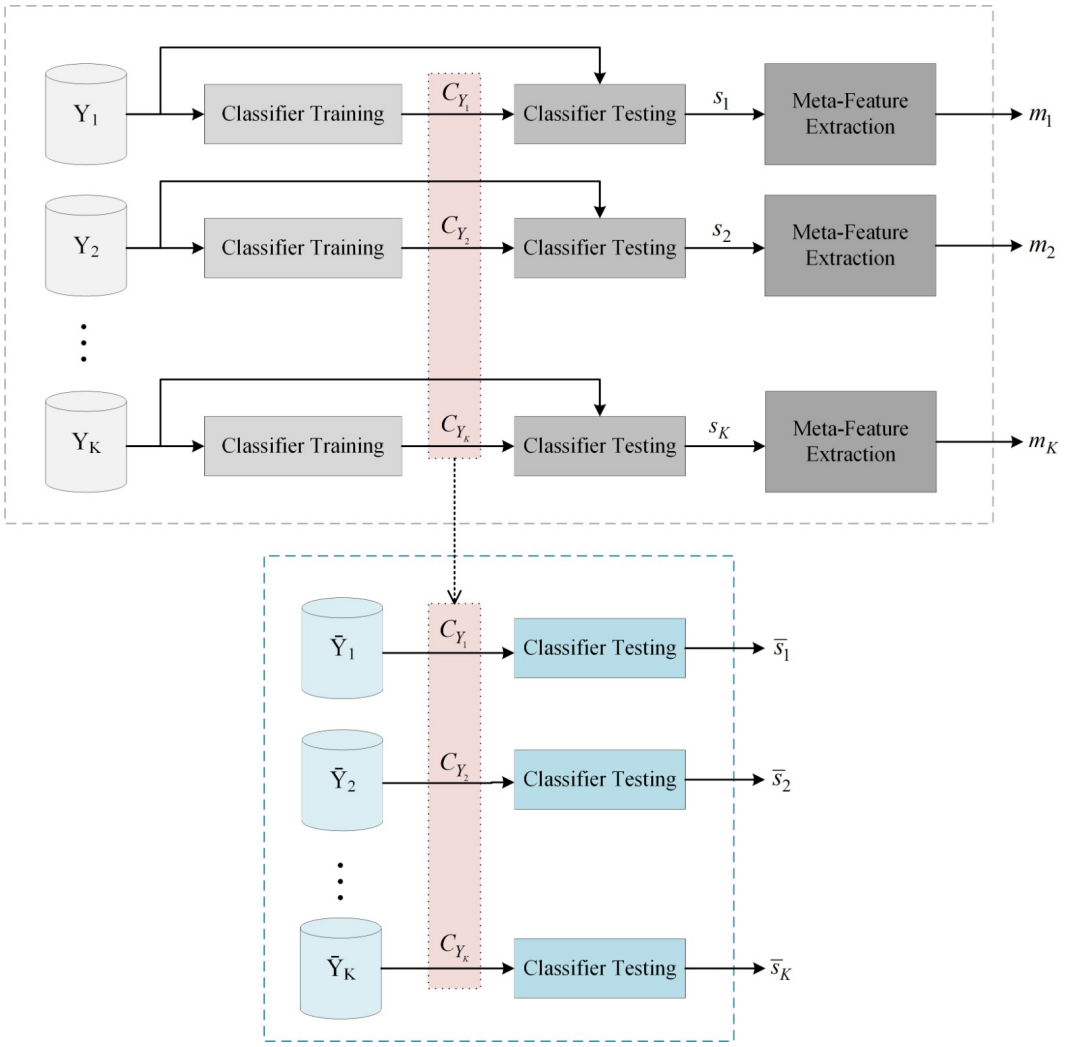


Figure 3.2: Extraction of meta-features using external datasets.

3.4 Extraction of Meta-Features

Consider a single-label binary classification task where each sample $x \in R^d$ is labeled as either positive or negative. Let X and $\overline{X}$ denote the training and the test sets, respectively. Assume that Y_k and $\overline{Y}_k$, $k = 1, \ldots, K$ denote that training and test splits of K different external datasets. For a given sample y in Y_k, the input space is of dimensionality d_k. Let M_k denote the total number of samples in Y_k. Assume that C_X represents a classifier trained using X. For a given $x \in X$, $C_X(x)$ corresponds to the probability score assigned to the positive class.

For each external dataset, a classifier denoted by C_{Y_k} is trained using the

corresponding training split, Y_k as illustrated in Figure 3.2. Then, by using the same split of the external dataset, the classifier is tested and the score vector, $s_k \in R^{M_k}$ is obtained. It should be noted that each element in the score vectors corresponds to the probability that the correct label is p. The elements of the score vectors are then sorted according to their values in descending order. The meta-feature vectors denoted by m_k are computed for all external datasets by using the sorted score vectors. Let the first positive be defined as the positive training sample having the largest score among all positives and the last positive is the one having the smallest score among all positives. The meta-features that are defined using s_k are presented in Table 3.1. To the best of our knowledge, the predictors other than **CntPos**, **ScrPosFrst**, **ScrPosLst**, **TrOptTrain** and **maxMetric** were not considered before for thresholding. In defining the meta-features, the ultimate goal was to extract information about the distribution of the positive samples in the sorted list. An ablation study is conducted to verify the performance of the proposed set by evaluating their performances individually and in subsets. The details are presented in Chapter 5.

Consider the example presented in Figure 3.3 which presents an s_k of $M_k = 14$ samples that are sorted in descending order. The true labels are provided to the left of the scores. The binning information of the samples between the first and the last positive is shown in terms of the bin indices on the right of the scores. The meta-feature vector computed is presented on the right. For instance, CntPos is computed as 5 since there totally five positives in the training set.

For each of the K external datasets, the classifier C_{Y_k} is trained by and tested on Y_k. Consequently, using m_k as the kth row, the meta-data $D \in R^{K \times F}$ is obtained where K is the total number of external datasets and $F = 17$.

Table 3.1: The proposed meta-features.

RatePtoN	The number of p-to-n transitions divided by the number of positives. This feature characterizes the distribution of the samples in the feature space. Ideally, the positives should appear as a single block, yielding the value $1/P$.
CntPos	The number of positive samples in the dataset.
CntNegFrst	The number of negative samples before the first positive. This feature evaluates the performance of the classifier in placing the positives on top ranks.
CntNegBtwn	The number of negatives between the first and the last positive. This feature is used to represent the overlap between positives and negatives.
ScrPosFrst	The probability score of the top-ranked positive. Since the best-fitting threshold depends on scores, this is used to represent the score of the first positive as a meta-feature.
ScrPosLst	The probability score of the last positive.
ScrSmplP	The average of the probability scores at Pth and $(P+1)$th probability scores. This feature represents the gap between the score of the lowest-ranked positive and the following negative. This gap represents the separation of different classes in score space. Ideally, the scores for positives are 1 and for the negatives, they are 0. Therefore, the ideal value for this meta-feature is 0.5.
AvgScrPtoN	The average of the probability scores at p-to-n transitions. This feature is used to identify the range of scores where positive to negative transitions occur.
AvgScrNtoP	The average of the probability scores at n-to-p transitions. This feature is used to identify the range of scores where negative to positive transitions occur.
BinRate#1, . . . , #5	The scores between the first and the last positive are discretized into 5 equally-spaced bins. From each bin, a meta-feature is extracted as the proportion of positive samples.
Gini	Gini impurity of the bins. This feature is used to measure the dispersion of the positive samples in different bins.
TrOptTrain	The best-fitting threshold that maximizes the performance metric on the training data. It is obtained by examining the p-to-n transitions and recording the threshold which maximizes the concerned metric.
maxMetric	The maximum metric value that is achieved at the optimal threshold denoted by TrOptTrain. For $F-score$, it is named as Fmax.

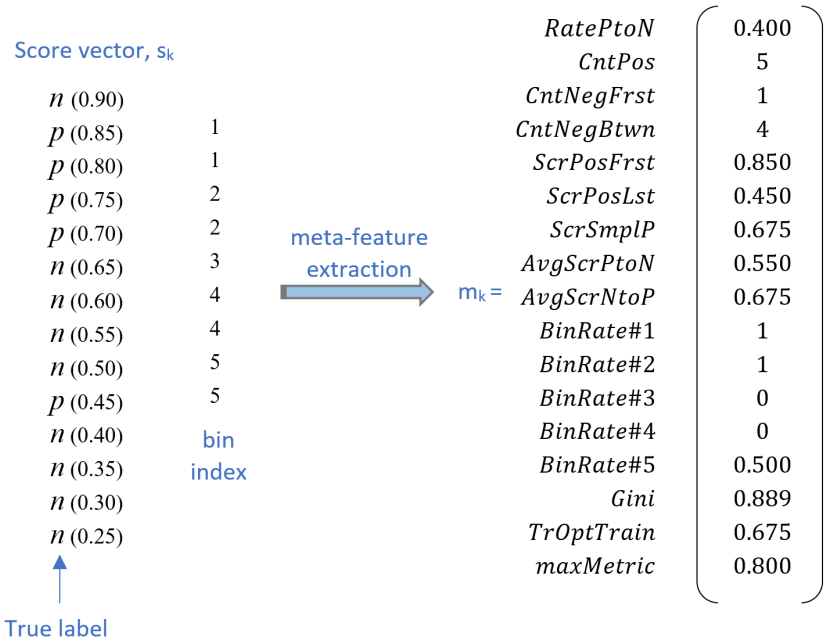


Figure 3.3: An example for meta-feature extraction.

In order to compute the target values, each C_{Y_k} is also tested using the corresponding test split, $\overline{Y}_k$ to compute the score vector $\overline{s}_k$, $k = 1, \ldots, K$ as shown in Figure 3.2. The threshold τ_k that provides the highest performance score on the sorted $\overline{s}_k$ is defined as the target threshold. Hence, as illustrated in Figure 3.4, the meta-data $D \in R^{K \times F}$ and the target threshold vector $\tau \in R^K$ are used to form the training set of the meta-learner, $T = \{(m_k, \tau_k) \,|\, k = 1, \ldots, K\}$.

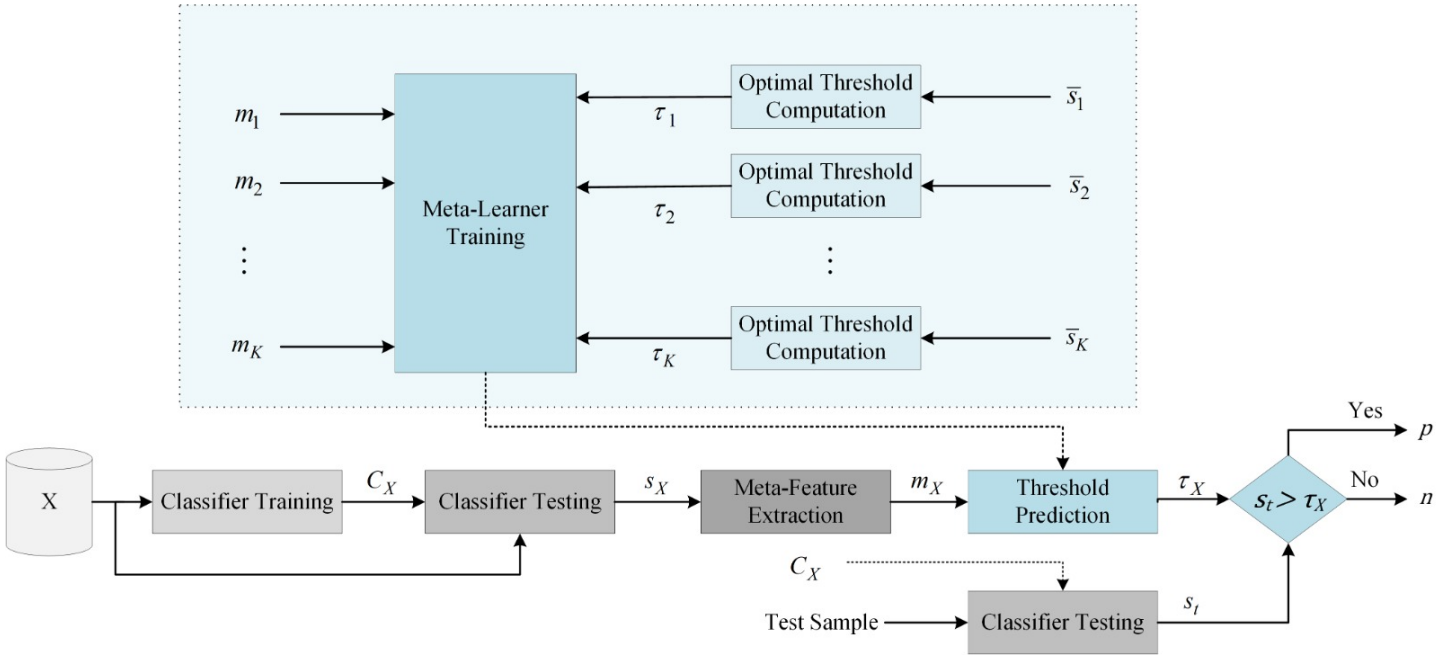


Figure 3.4: Training a meta-learner and its use during testing.

3.5 Training the Meta-Learner

Training the meta-learner is an offline task. In this thesis, random forest (RF) is selected as the meta-learner. The motivation for using RF is twofold. Firstly, although mapped to the meta-space of the same dimensionality, external dataset have different characteristics in terms of imbalance ratio, number of samples, degree of imbalance and the dimensionality of feature vectors. Since an RF is made up of multiple decision trees, it can be considered as a flexible architecture that can effectively represent information from different sources. Another strength of RF is its implicit feature selection. Due to the rarity of the positive class and hence the risk of overfitting, choosing the best feature subset by considering the relevance and redundancy of the meta-features is not feasible. Therefore, it is expected that the training algorithm of RF will choose a good subset that generalizes well for unseen datasets.

During testing an unseen dataset, the following procedure is applied. Using the training set of the dataset denoted by X, a classifier is firstly built. Then, the classifier is tested using the training split to compute the score vector, s_X. After computing the meta-feature vector m_X from s_X, the meta-learner is used to predict the optimal threshold denoted by τ_X for the dataset. The decision whether a test sample is positive or not is made by comparing the probability score produced by the classifier, s_t with the predicted threshold, as shown at the bottom of Figure 3.4.

The proposed meta-learner is also tested when ensembles are used. Figure 3.5 illustrates the "Ensemble Training" and "Ensemble Testing" blocks used to replace "Classifier Training" and "Classifier Testing" blocks in Figures 3.2 and 3.4. In the experiments conducted, homogeneous classifier ensembles of 100 members are

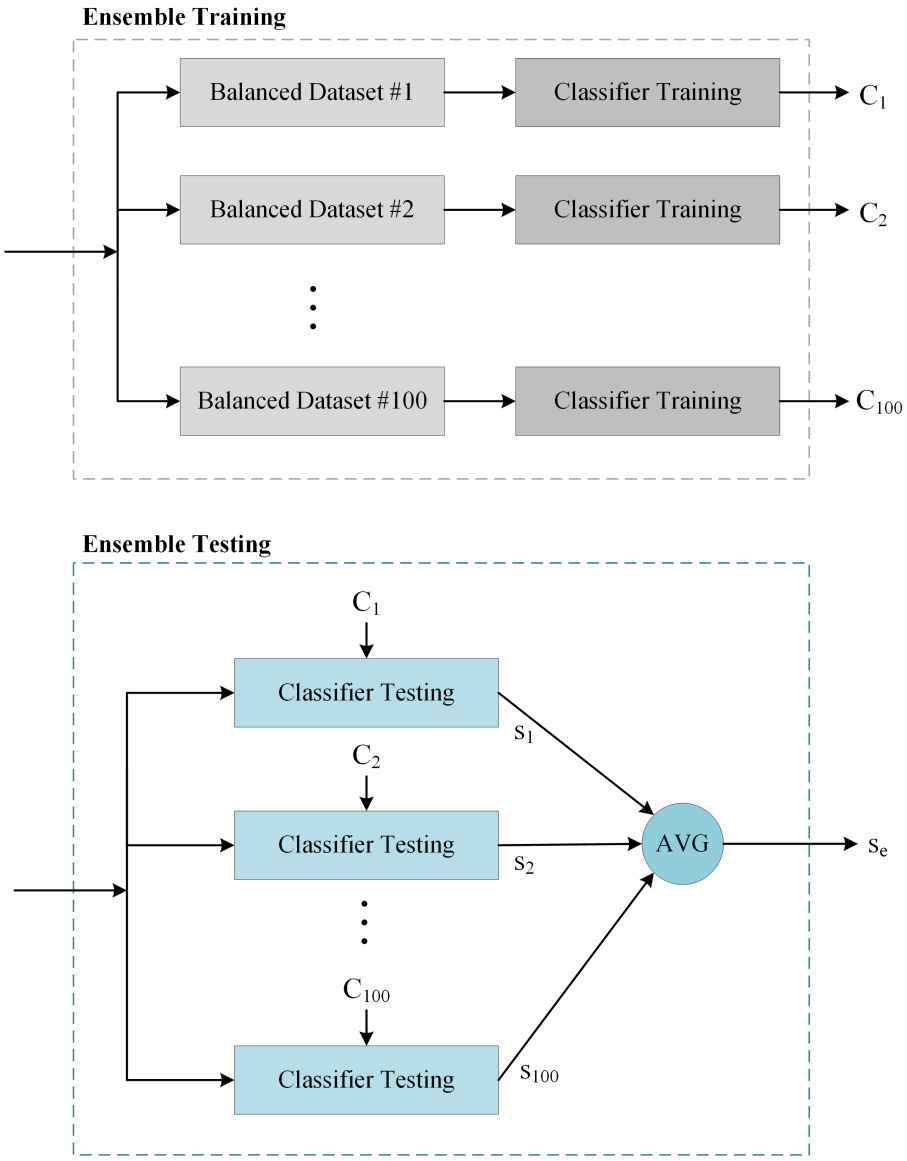


Figure 3.5: Blocks to replace the classifier training and testing blocks in Figures 3.2 and 3.4 by ensembles.

utilized for each balancing method considered. In other words, for a given balancing method and a given training split, 100 balanced data sets are generated and accordingly, 100 classifier models are trained. The classification scores of the specified training split are obtained by averaging the scores resulting from the trained models. The meta-features are then computed using the averaged scores. Similarly, the classification scores of the corresponding test split are computed as the average of the scores attained by testing the aforementioned models on the test split.

Chapter 4

IMBALANCE LEARNING BY SAMPLE REPOSITIONING

4.1 Introduction

The ultimate goal of oversampling is to enforce the learner to label the feature space including minority samples as the positive class, although some of these regions might be outnumbered by the negative samples. In fact, this can be achieved using an alternative approach. Rather than increasing the numbers of positive samples in the subspace where they appear, we propose to *reposition* the negative samples towards their centroid so as to reduce the number of majority samples in the subspace where the minority samples exist. Similarly, the positives are repositioned but in small steps to effectively deal with different types of minority samples such as borderline or outlying instances. This approach has several advantages compared to the oversampling-based approaches. For instance, small displacements will keep the positive borderline samples close to their original positions and hence do not distort the boundary. After repositioning, the outliers that are more likely to be in the majority class space will not mislead the learner as their original forms since they will be closer to the other positives. Moreover, irrelevant positive regions are not generated since the minority instances are slightly modified. Consequently, the assumptions made by some classifiers about the original class distribution such as the normality of feature values are not violated. The hard-to-classify minority samples problem is also addressed by reducing the number of negatives in the corresponding

spaces.

The first step of the proposed MAjority and MInority rePOsitioning Technique (MaMiPot) is to train the learner on the given dataset to identify the misclassified samples, that is the set of false positives (sFP) and the set of false negatives (sFN). The initial set of the seed samples is computed as the union of sFP and sFN. The algorithm repositions the seed samples in sFP using the centroid of the true negatives by linear interpolation. Similarly, the samples in sFN are repositioned by employing the centroid of the correctly classified positive samples. The learner is re-trained by employing the updated training set and evaluated on the original training data to check for improvement in the performance metric. The procedure is repeated until further performance gain is not achieved.

In order to understand the main idea behind this approach, consider the toy example given in Figure 4.1. Due to class imbalance, most of the positives (represented by red diamonds) are expected to be misclassified, as shown in the left part of the figure. In other words, the decision region for the minority class covers a much smaller space than the actual space where the positive samples appear. Such solutions generally lead to low classification accuracy on the positive samples (i.e. sensitivity) but high accuracy on the negatives (i.e. specificity). As shown in the upper right part of the figure, the solution by SMOTE is to generate synthetic minority samples to enlarge the positive decision region, leading to a more acceptable solution. As an alternative approach, the idea behind MaMiPot is presented in the lower right part of the figure. The positive and negative samples are repositioned for the same purpose of enlarging the minority decision region. The repositioned instances are shown using the same symbol of the class but not filled with the corresponding color. Since the majority

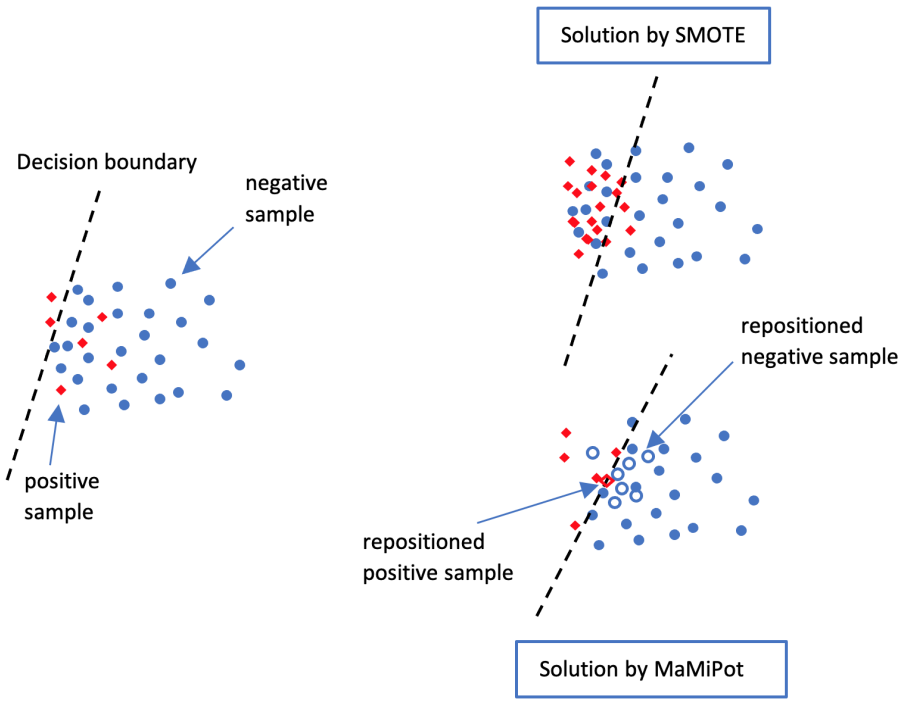


Figure 4.1: Comparison of SMOTE and MaMiPot on a toy example.

samples overlapping with the minority are moved away from the minority region, minimizing the classification accuracy will correspond to labeling a wider space as positive. Moreover, since the distribution of minority samples is not distorted, the borderline is computed more accurately. Although the motivation behind moving the majority samples is intuitively reasonable, repositioning the minority instances needs further clarification. As mentioned above, one of the main challenges in SMOTE is outliers. These samples may lead to generating synthetic instances in the majority class region and hence severely mislead the learner. To alleviate this problem, the proposed approach repositions the minority samples as well. However, in order not to distort the distribution of the minority samples, the amount of repositioning is set to be in smaller steps for the minority class.

There are two main issues that need to be addressed for the implementation of the idea. These are the selection of the samples and the directions to which they will be moved.

In MaMiPot, the decision boundary computed on the original training data is employed for guiding the selection and repositioning tasks. The details of the proposed method are presented in Algorithm 1. The first step of the algorithm is to train a classifier. In Step 2, the classifier is tested using the training data. The next step, Step 3 is to obtain the decision boundary by computing the best-fitting decision threshold, τ_{opt}. The performance metric is a design parameter of the algorithm. Hence, τ_{opt} should be defined by taking into account the metric, M. Using the selected threshold, the value of the metric, M_{opt} is recorded. Selection of τ_{opt} is explained in detail in the following context.

Using the decision boundary obtained, the training samples are partitioned into four sets. sTP denotes the set of correctly classified minority samples (i.e. true positives). sFN represents the set of misclassified minority samples (i.e. false negatives). The set sTN includes correctly classified majority samples (i.e. true negatives). sFP denotes the set of misclassified majority samples (i.e. false positives). The next step is to compute the centroids of the correctly classified minority samples denoted by μ_P and the correctly classified majority samples denoted by μ_N.

The samples in sFN and sFP are the candidates to be repositioned. In the first part of the iterations starting on line 11, the misclassified minority samples in sFN are repositioned towards the centroid of the correctly classified minority samples using the positive repositioning factor, α_P. $\widehat{sFN}$ is computed as the updated set after repositioning. In this thesis, we assumed that the correctly classified samples form a single cluster.

It should be noted that, at this step only the locations in the set sFN changed. As a

Algorithm 1: MaMiPot

Input: T: Training data (P minority and N majority samples), C: Classifier, M: Performance metric, α_P: Positive (minority) repositioning factor, α_N: Negative (majority) repositioning factor, γ: cluster size threshold

Train the classifier C using the training data T
Test C using T
Compute the threshold, $\tau_{opt} = \frac{1}{2}\left(\frac{P}{P+N} + 0.5\right)$ and record the value of M computed using τ_{opt}, denoted by M_{opt}
Compute the sets sTP, sFP, sTN and sFN using τ_{opt}
Calculate the centroid of samples in sTP, denoted by μ_P
Calculate the centroid of samples in sTN, denoted by μ_N
$iter \leftarrow 0$, $T_{opt} \leftarrow T$
while $iter < 3$ **do**
    $iter \leftarrow iter + 1$
    //Repositioning minority samples in sFN:
    if $|sTP| > \gamma \quad \& \quad |sFN| > 0$ **then**
        for *each* x_i *in* sFN **do**
            Move x_i towards μ_P using α_P:
            $\hat{x}_i = \alpha_P * \mu_P + (1 - \alpha_P) * x_i$
        end for
    end if
    //Repositioning majority samples in sFP:
    if $|sTN| > \gamma \quad \& \quad |sFP| > 0$ **then**
        for *each* y_i *in* sFP **do**
            Move y_i towards μ_N using α_N:
            $\hat{y}_i = \alpha_N * \mu_N + (1 - \alpha_N) * y_i$
        end for
    end if
    Define T_{new} as the union of sTP, sTN and the repositioned sets $\widehat{sFN}$ and $\widehat{sFP}$
    Train the classifier C_{new} using T_{new}
    Test C_{new} using the original training data T and, using τ_{opt} record M_{new}
    if $M_{new} > M_{opt}$ **then**
        $M_{opt} = M_{new}$
        $T_{opt} = T_{new}$
        $iter \leftarrow 0$
    end if
end while

Output: The repositioned training set, T_{opt}

condition for making the repositioning, we check whether the number of correctly classified samples is above a threshold denoted by γ. The motivation for this condition can be explained as follows. The centroid is computed using only the correctly classified minority samples. In MaMiPot algorithm, the centroids are used as the representatives for the potential regions to be enlarged. By moving samples towards these points, it is aimed to enforce the learner to focus on that part of the feature space. Because of this, for a centroid to convey a potential for the learner to discover additional minority regions, a predefined number of samples denoted by γ are required to be already correctly classified. For instance, if all minority samples are misclassified, none of them is repositioned.

In the second part of the iterations starting on line 16, the misclassified majority samples are repositioned towards the centroid of correctly classified majority samples using the negative repositioning factor, α_N. The new positions are recorded, obtaining the updated set, $\widehat{sFP}$. As in the case of minority samples, we check whether the number of correctly classified majority samples is above the threshold, γ. The new training set, T_{new} containing the correctly classified samples (i.e. sTP and sTN) and, repositioned samples (i.e. $\widehat{sFN}$ and $\widehat{sFP}$), is used to construct a new classifier, C_{new} as presented on line 21. This classifier is tested on the original training set, T to evaluate whether the performance is improved. If the performance is improved, T_{new} is recorded as T_{opt} and the iterations are repeated.

During the iterations, the centroids are not updated. The main reason can be explained as follows: The ultimate goal of the algorithm is to reposition the samples but not the class. Allowing changes in the direction of repositioning may lead to translation of the classes in the feature space which is not desired. Actually, such operation will not be

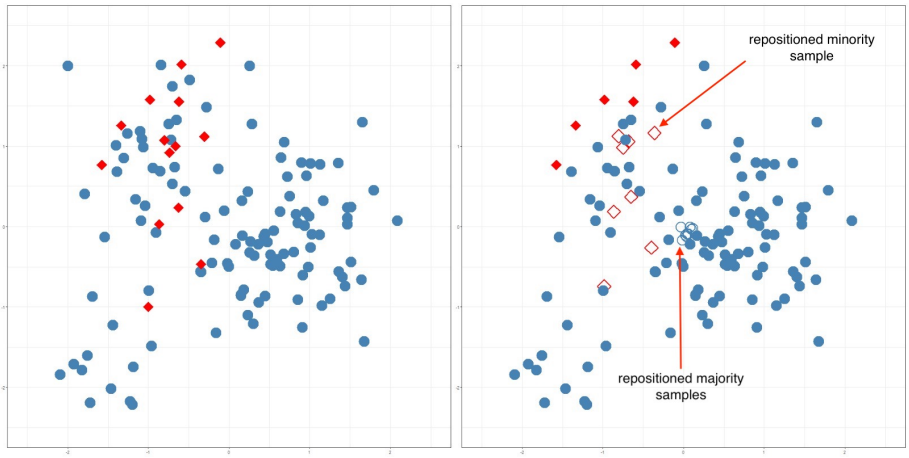


Figure 4.2: Repositioning of positive and negative samples using MaMiPot. The design parameters were set as $\alpha_N = 0.9$ and $\alpha_P = 0.1$.

rewarded since the repositioning done in each iteration is evaluated using the original training set on Line 22 of the algorithm.

The repositioning factors α_P and α_N represent the percentage of repositioning on the lines connecting the minority and majority instances, respectively. In order not to distort the distribution of minority samples, α_P should be kept small. In other words, the minority samples must be repositioned in small steps. On the other hand, larger displacements should be allowed by employing large α_N values. In fact, α_P is mainly used to deal with the outliers. During the iterations, they are moved towards the centroid computed for the minority class. Figure 4.2 presents the execution of MaMiPot, where $\alpha_N = 0.9$ and $\alpha_P = 0.1$. The figure on the left shows the dataset. The minority samples are presented using the symbol '$\diamond$' filled by red color. The repositioned instances are plotted using non-filled symbols. As it can be seen, the positive and negative samples in the overlapping regions are moved towards the corresponding centroids. Consequently, the classifier is expected to increase the number of true positives. It can be also be seen that the minority samples are moved slightly.

If three consecutive updates do not lead to further performance improvement, the iterations are terminated and T_{opt} is returned as the best-fitting training set. Remember that the centroids μ_P and μ_N are kept fixed during the iterations. Since α_N is selected as a large value, after the first iteration, the repositioned negatives will be close to μ_P. As a matter of fact, the remaining room for repositioning is small for the majority samples after the first iteration. On the other hand, since the minority samples are repositioned in small steps, they can continue to reposition as the number of iterations increase. Consequently, the first priority of the algorithm is to solve the problems due to imbalance by repositioning the negatives. If a performance gain is not achieved any more, by running for two more iterations, MaMiPot tries the lower priority option and targets at achieving further improvements by repositioning the minority samples. When $\alpha_P = 0.1$, the distance between a minority sample and its centroid will be reduced by $1-(0.9)^3 = 0.27$ (i.e. 27%) after three iterations which is much less than the repositioning of majority samples in the first iteration (i.e. 0.90). This is consistent with our main target of avoiding altering the distribution of positive samples. Alternative values for the number of iterations can be empirically evaluated for the task under concern.

The performance metric is another design parameter of the algorithm. Accuracy is not an appropriate measure in imbalance learning. Since the number of minority samples is only a few in general, very high accuracy values can be obtained even by classifying all samples as negative. Alternatively, F-score, G-mean and AUC can be considered. AUC is the area under the receiver operating characteristic curve. However, AUC is not threshold dependent. As mentioned above, the main idea of MaMiPot is to reposition the misclassified minority and majority samples. As a matter of fact, AUC can be used as a design parameter only if the samples to be repositioned are determined using a

heuristic approach. For instance, majority samples that appear in the highest P ranks and minority samples appearing in the lowest N ranks can be selected. Alternatively, as done in several variants of SMOTE, the neighborhoods of the minority samples or the density of the minority samples in different clusters can be considered. On the other hand, F-score or G-mean can be employed directly since they are threshold-dependent. In this thesis, in order not to bias the results by re-running the algorithm for all metrics, the algorithm is run only for F-score, and the performance scores of all three metrics are reported. In a recent study, the following threshold is proposed for F-score [43]:

$$\tau_{opt} = \frac{1}{2}\left(\frac{P}{P+N} + 0.5\right) \tag{4.1}$$

where 0.5 is the maximum possible threshold value for maximizing F-score [42] and $P/(P+N)$ is the a-priori probability of the minority class. The midway between these two term is proposed as the threshold for F-score. In our simulations, we used this threshold value.

In general, since the minority samples are outnumbered by the majority, we expect low recall values [106]. In other words, the classification performance on the positive samples is generally low. The main reason for this is the bias in favor of the majority class samples. On the other hand, precision will be higher when compared to employing a balanced dataset. This can be explained as follows: After balancing, the decision region for the minority enlarges, including more correctly classified minority samples and hence higher recall. However, the number of false positives will also increase. Due to the imbalance, we expect a larger increase in FP than TP, leading to a smaller precision score. Let the *balancing factor,* β be defined as the proportion of the synthetic samples to the difference in the class sizes, $\beta = N_{syn}/(N_{maj} - N_{min})$ where N_{syn} denotes the number of synthetic samples that are generated in the case of

N_{maj} majority and N_{min} minority instances. In order to tackle the low recall problem, SMOTE oversamples with balancing factor equal to one whereas MaMiPot aims to alleviate the imbalance problem by moving the negatives away from the positives. When the minority class is small, if the performance metric is selected as recall, a large degradation in precision may occur. In order to avoid this, the algorithm should be run using a trade-off metric such as F_1 and G-mean.

For a better understanding of how MaMiPot will contribute the performance scores for different metrics, both class-imbalance and characteristics of the metrics should be taken into consideration. As it was emphasized in the recent study by Soleymani *at al.* [107], both AUC and G-mean favor increasing the number of true positives, even for much higher number of misclassified instances from the majority class. This can be understood by considering the toy example presented in Figure 4.3. Assume that there are 10 minority and 100 test samples. Using a learner that is trained on the training set which is not balanced, let the number of true positives and true negatives be 5 and 90, respectively. After oversampling the minority class, we expect a higher true positive value. However, this can be achieved at the expense of true negatives. Let the number of true positives be increased by only one whereas the number of true negatives is decreased by 5. This corresponds an increase in sensitivity from $5/10$ to $6/10$ and a decreased specificity from $90/100$ to $85/100$. When compared to employing the original training data, although the total number of misclassified samples increased by $5-1=4$, G-mean increases from 0.67 to 0.71. Hence, G-mean favors increasing the number of true positives.

MaMiPot improves the performance on the minority class mainly by repositioning the majority samples. However, this is done in a global way. In particular, the algorithm

using imbalanced training set

actual \ predicted	P	N
P	5	5
N	10	90

precision	5/15
recall	5/10
Specificity	90/100
F-score	0,40
G-mean	0,67

using balanced training set

actual \ predicted	P	N
P	6	4
N	15	85

precision	6/21
recall	6/10
Specificity	85/100
F-score	0,39
G-mean	0,71

Figure 4.3: An example to illustrate the effect of balancing in terms of the metric values.

does not focus on any specific region by evaluating the clustering of samples. Similarly, in SMOTE, the whole feature space of the minority class benefits from oversampling. On the other hand, in some variants of SMOTE, oversampling is applied in a selected subspace. For instance, kmeans-SMOTE considers the density of minority samples to select clusters to be oversampled. Since MaMiPot is not cluster or region-based, some minority samples may not benefit from repositioning. Oversampling the minority class has the potential to be a complementary tool for such samples. In this setting, MaMiPot will be utilized as a preprocessing step before oversampling, providing several advantages for the oversampler. For instance, due to repositioning of the majority samples by MaMiPot, a smaller balancing factor than one is expected to be better-fitting, leading to less distortion on the minority distribution when compared to SMOTE and most of its variants. Moreover, utilizing a small balancing factor will result in a similar imbalance for both training and test data, which is another important concern in imbalance learning [103]. By repositioning outlying minority samples towards the centroid of the correctly classified instances, the risk of generating synthetic samples in invalid regions is reduced.

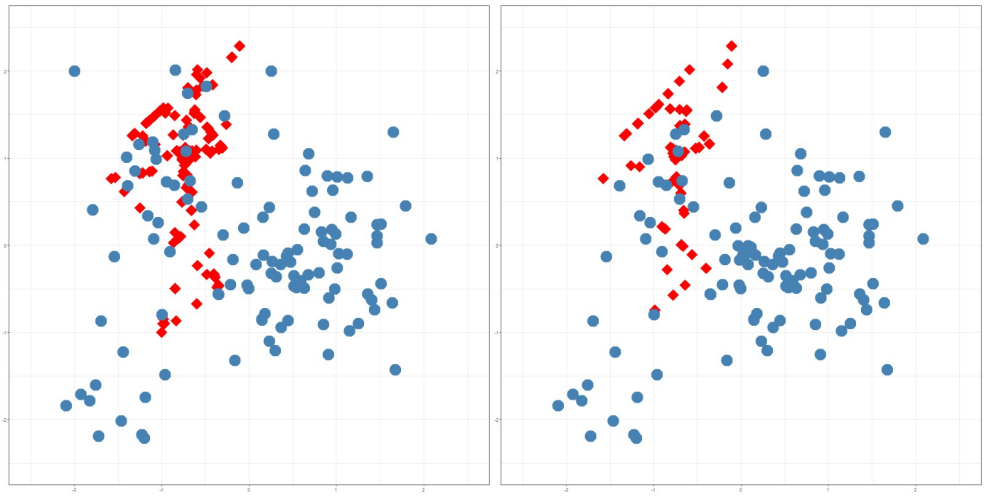


Figure 4.4: Balancing the data using SMOTE (left figure) and balancing positive samples after repositioning by MaMiPot and using $\beta = 0.5$ (right figure). The design parameters of MaMiPot were set as $\alpha_N = 0.9$ and $\alpha_P = 0.1$.

Precision replaces the specificity in the case of F-score [107]. Since the minority class includes small number of samples in general, if the increase in true positive rate due oversampling is achieved at the expense of precision due to increased number of false positives, F-score value will degrade. As shown in Figure 4.3, F-score decreases from 0.40 to 0.39. Consequently, when F-score is the performance metric, SMOTE is expected to contribute MaMiPot only if it is run using a small balancing factor. The left part of Figure 4.4 presents the execution of SMOTE on the data given in Figure 4.2. The dataset obtained after running MaMiPot using $\alpha_N = 0.9$ and $\alpha_P = 0.1$, followed by oversampling using SMOTE for $\beta = 0.5$ is illustrated on the right. It can be seen in the figure that, since many majority samples are repositioned, a smaller number of synthetic samples is expected to be sufficient for achieving a reasonable recall value.

Chapter 5

EXPERIMENTAL WORK

5.1 Introduction

The experiments are conducted on 52 imbalanced data sets from KEEL collection [108]. The datasets are listed in Table 5.1. This collection consists of two-class datasets that are computed from multi-class data sets. The imbalance ratios (IR) of the selected datasets, i.e. the ratios of negatives to positives, are in the range 5.14 to 129.44. The number of positive samples and the imbalance ratios are also presented in the table. As can be seen in the table, positive samples are highly rare in most of the datasets. In our simulations, 5×2-fold cross-validation is applied, obtaining 10 folds per dataset.

5.2 Experiments on Evaluating the Meta-Learner Approach

The proposed threshold predictor model is assessed by a comprehensive set of experiments using a linear and a nonlinear classifier, namely logistic regression (LR) and support vector machine (SVM) with radial basis function kernel. These classifiers are among the most widely-used classification methods in imbalance learning. The computational cost of SVM is high. Based on the empirical evidence from preliminary experiments, we fixed its parameters as $\sigma = 0.25$ and $C = 0.25$ in all simulations. During testing, the scores generated by the classifiers represent the probability that the corresponding sample belongs to the positive class. As a matter of fact, correctly classified negative samples are assigned a score that is less than 0.5. There are a total of 520 folds in the experiments. Hence, the value of K in Figure 3.2

Table 5.1: The datasets selected from KEEL repository.

Rank	dataset	#Positives	IR
1	abalone19	32	129.44
2	yeast6	35	41.40
3	ecoli-0-1-3-7_vs_2-6	7	39.14
4	yeast5	44	32.73
5	yeast-1-2-8-9_vs_7	30	30.57
6	yeast4	51	28.10
7	yeast-2_vs_8	20	23.10
8	glass5	9	22.78
9	yeast-1-4-5-8_vs_7	30	22.10
10	shuttle-c2-vs-c4	6	20.50
11	glass-0-1-6_vs_5	9	19.44
12	abalone9-18	42	16.40
13	page-blocks-1-3_vs_4	28	15.86
14	ecoli4	20	15.80
15	glass4	13	15.46
16	yeast-1_vs_7	30	14.30
17	shuttle-c0-vs-c4	123	13.87
18	ecoli-0-1-4-6_vs_5	20	13.00
19	cleveland-0_vs_4	13	12.62
20	ecoli-0-1-4-7_vs_5-6	25	12.28
21	glass2	17	11.59
22	glass-0-1-4-6_vs_2	17	11.06
23	ecoli-0-1_vs_5	20	11.00
24	glass-0-6_vs_5	9	11.00
25	led7digit-0-2-4-5-6-7-8-9_vs_1	37	10.97
26	ecoli-0-1-4-7_vs_2-3-5-6	29	10.59
27	glass-0-1-6_vs_2	17	10.29
28	ecoli-0-6-7_vs_5	20	10.00
29	vowel0	90	9.98
30	yeast-0-5-6-7-9_vs_4	51	9.35
31	ecoli-0-3-4-7_vs_5-6	25	9.28
32	ecoli-0-3-4-6_vs_5	20	9.25
33	glass-0-4_vs_5	9	9.22
34	ecoli-0-2-6-7_vs_3-5	22	9.18
35	ecoli-0-1_vs_2-3-5	24	9.17
36	ecoli-0-4-6_vs_5	20	9.15
37	yeast-0-2-5-6_vs_3-7-8-9	99	9.14
38	yeast-0-2-5-7-9_vs_3-6-8	99	9.14
39	yeast-0-3-5-9_vs_7-8	50	9.12
40	glass-0-1-5_vs_2	17	9.12
41	ecoli-0-2-3-4_vs_5	20	9.10
42	ecoli-0-6-7_vs_3-5	22	9.09
43	yeast-2_vs_4	51	9.08
44	ecoli-0-3-4_vs_5	20	9.00
45	page-blocks0	559	8.79
46	ecoli3	35	8.60
47	yeast3	163	8.10
48	glass6	29	6.38
49	segment0	329	6.02
50	ecoli2	52	5.46
51	new-thyroid2	35	5.14
52	new-thyroid1	35	5.14

is $51 \times 10 = 510$. The experiments are conducted through leave-one-dataset-out by keeping the *test* dataset out and exploiting the remaining datasets as the *external* datasets for training the meta-learner. The meta-features are firstly extracted from the classification scores of the external training splits and then the target thresholds, those which result in the highest $F-score$, are obtained from the associated test splits. The meta-learner learns a generic mapping from the meta-features to the optimal thresholds. Finally, the model is tested on the meta-feature vector extracted from the training split of the unseen test data to predict the threshold. The same model is used for all 10 folds of the data set that is held out.

To examine the effectiveness of the proposed threshold prediction method, the meta-leaner approach is first compared with six threshold-moving methods suggested in the literature. These methods are summarized in Table 5.2, along with their computational complexities. The big $\mathcal{O}$ notations depict how the execution time for threshold calculation grows during the testing phase as the number of samples in the training set increases. To make a fair comparison regardless of the classification model, we assume that prediction scores for the training samples are already obtained. In the table, P and N are the numbers of positive and negative samples in the training set, and M is an internal variable in SVM-OTHR algorithm referring to the number of misclassified minority samples, where $M \leq P$ [41]. optTrain and FDiv2 entail $F-score$ calculation which is in $\mathcal{O}(P+N)$. Counting occurrences for Pmin, MID and SVM-THR is also in $\mathcal{O}(P+N)$. The SVM-OTHR algorithm comprises several steps. It starts with $G-mean$ calculation of the training set ($\mathcal{O}(P+N)$), finding misclassified positive samples ($\mathcal{O}(P)$) and ranking them ($\mathcal{O}(MlogM)$). Then, it iterates over all the M samples by finding their closest neighbor from the negative class ($\mathcal{O}(MN)$), computing the adjusted distance ($\mathcal{O}(M)$), and finding the $G-mean$ of the training set

for the new threshold ($\mathcal{O}(M(P+N))$). Finally, the threshold corresponding to the maximum $G-mean$ is found ($\mathcal{O}(M)$). Putting all of these together, SVM-OTHR is in $\mathcal{O}(P(P+N))$. The complexity of the proposed approach is evaluated by considering two fundamental stages in the testing phase: calculation of the feature vector given the classification scores of the training set and applying the trained RF to obtain the optimal threshold. The most demanding computations in feature extraction are counting occurrences and ranking which results in $\mathcal{O}(P+N)$ being the dominating term in the computational complexity analysis. Testing the RF meta-learner is in $\mathcal{O}(D*T)$ where D is the depth of a tree and T is the number of trees. Thus, the proposed algorithm is in $\mathcal{O}(P+N)+\mathcal{O}(D*T)$. Considering the fact that $\mathcal{O}(P+N)$ grows linearly with the size of the input and $\mathcal{O}(D*T)$ is constant once the meta-learner is trained using external data sets, the proposed approach is in $\mathcal{O}(P+N)$.

To provide further evidence of the effectiveness of the proposed method, the performance of the meta-learner threshold predictor is compared with five well-known balancing techniques namely, random undersampling, random oversampling, SMOTE, ADASYN, and DBSMOTE. Two recent balancing techniques, GSMOTE and SMOTEFUNA are also considered.

In the first set of experiments, the proposed meta-learner is inspected on simple classifiers without balancing to predict the threshold for the test split of the unseen data sets. Table 5.3 shows the $F-score$s obtained by the meta-learner and four competing threshold-moving methods for LR and SVM. It should be noted that SVM-THR is proposed for SVM and it is not applicable to LR. Average scores across both classifiers are also given in the last row. The highest scores in each row are marked in boldface.

Table 5.2: Competing threshold-moving methods.

Method	Description	Complexity	Source
optTrain	Threshold that maximizes $F-score$ on train split	$\mathcal{O}(P+N)$	[55]
Pmin	A priori probability of the minority class, $(\frac{P}{P+N})$	$\mathcal{O}(P+N)$	[62]
MID	The halfway between Pmin and 0.5, $(\frac{Pmin+0.5}{2})$	$\mathcal{O}(P+N)$	[43]
FDiv2	Maximum $F-score$ on train split divided by 2, $(\frac{Fmax}{2})$	$\mathcal{O}(P+N)$	[42]
SVM-THR	$(\frac{P-N}{P+N+2a})$, by default $a=1$	$\mathcal{O}(P+N)$	[60]
SVM-OTHR	Exhaustive search by considering misclassified minority samples	$\mathcal{O}(P^2+PN)$	[41]

It is inferred from the table that the proposed approach is superior to the alternative threshold-moving techniques for both classifiers. optTrain and MID are the nearest competitor for LR and SVM respectively. When compared in terms of average performance scores, the meta-learner is followed by MID, improving MID's average $F \quad score$ from 0·6378 to 0·6526. It is also observed that Pmin is the least effective threshold-moving method for imbalance classification.

The second set of experiments is conducted to evaluate the proposed approach when accompanied by balancing methods. In particular, the thresholds predicted by the meta-learner are used for the classifier ensembles including 100 members, where each member is trained on a balanced form of the training set. Table 5.4 depicts the $F \quad score$s obtained by applying only balancing and employing the default threshold of 0.5 in addition to the combination of balancing and threshold moving by the meta-learner. The highest $F \quad score$ in each row is shown in boldface. The underlined

Table 5.3: $F-scores$ obtained by threshold-moving without balancing.

	LR	SVM	Average
optTrain	0.6370	0.6251	0.6310
Pmin	0.5795	0.5765	0.5780
MID	0.6321	0.6435	0.6378
FDiv2	0.6362	0.6303	0.6333
SVM-THR	-	0.5790	0.5790
SVM-OTHR	0.6218	0.6078	0.6148
Meta-learner	**0.6414**	**0.6639**	**0.6526**

numbers indicate the best scores achieved for each classifier on average. For LR, when the threshold is set to 0.5, DBSMOTE results in the highest $F\ \ score$ which is 0.6157. Moving the threshold by the meta-learner improves this score by almost 3%. This is slightly lower than the $F\ \ score$ attained by only employing the meta-learner without balancing according to Table 5.3. For SVM, GSMOTE is the top balancing scheme when the default threshold is used. Moreover, jointly utilizing GSMOTE and the meta-learner provides the highest $F\ \ score$.

Table 5.3 illustrated that the meta-learner alone results in a $F\ \ score = 0.6639$ for SVM. A similar pattern appears in respect of average $F\ \ score$. This evidence indicates the following facts in imbalance classification problems: 1) The proposed approach is more successful than alternative threshold-moving methods; 2) Balancing methods traditionally practiced in imbalance learning can benefit from the proposed threshold predictor; 3) The proposed meta-learner on its own is more effective than balancing.

To further analyze the performance of the meta-learner in threshold prediction, the predicted thresholds are plotted versus the optimal thresholds in Figures 5.1 and 5.2 for the two classifiers with and without balancing. SMOTE is taken into account as

Table 5.4: $F-scores$ obtained by balancing with and without threshold-moving.

	LR		SVM		Average	
	0.5	Meta-learner	0.5	Meta-learner	0.5	Meta-learner
Undersampling	0.5078	0.6224	0.4821	0.5659	0.4949	0.5941
Oversampling	0.5835	0.6287	0.6196	0.6340	0.6016	0.6313
SMOTE	0.6025	0.6384	0.6244	0.6363	0.6135	0.6374
ADASYN	0.5793	0.6313	0.5980	0.6140	0.5887	0.6226
DBSMOTE	**0.6157**	**0.6407**	0.6261	0.6320	**0.6209**	0.6364
GSMOTE	0.5931	0.6355	**0.6301**	**0.6489**	0.6116	**0.6422**
SMOTEFUNA	0.6136	0.6281	0.6136	0.6266	0.6136	0.6274

an instance of commonly used balancing methods. Accordingly, the columns from left to right across Figure 5.1 correspond to LR and SVM. The columns from left to right across Figure 5.2 correspond to LR-SMOTE and SVM-SMOTE. TrOptTest, shown on the vertical axis is the threshold that maximizes the $F \quad score$ on the test split of the unseen data. The horizontal axis in each subplot refers to the threshold computed by one of the top three threshold-moving methods listed in Table 5.2, namely optTrain, MID and FDiv2. Note that the diagonal line corresponds to the ultimate desirable case in adjusting the threshold. The mean square errors in computing thresholds for different schemes are also presented in the figures.

The salient trait observed in the figures is that the optimal threshold has a wide range spanning from 0 to 1. Meta-learner and optTrain are relatively successful in adjusting this wide-ranging threshold. However, the thresholds computed by MID and FDiv2 lie in a narrow range. Regarding the mean square error, the meta-learner results in a remarkably lower mean square error for SVM, LR-SMOTE, and SVM-SMOTE compared with the other three alternatives. For the case of LR presented in the leftmost column, although the lowest error is achieved by FDiv2, it is verified that the highest $F \quad score$ is attained by the meta-learner.

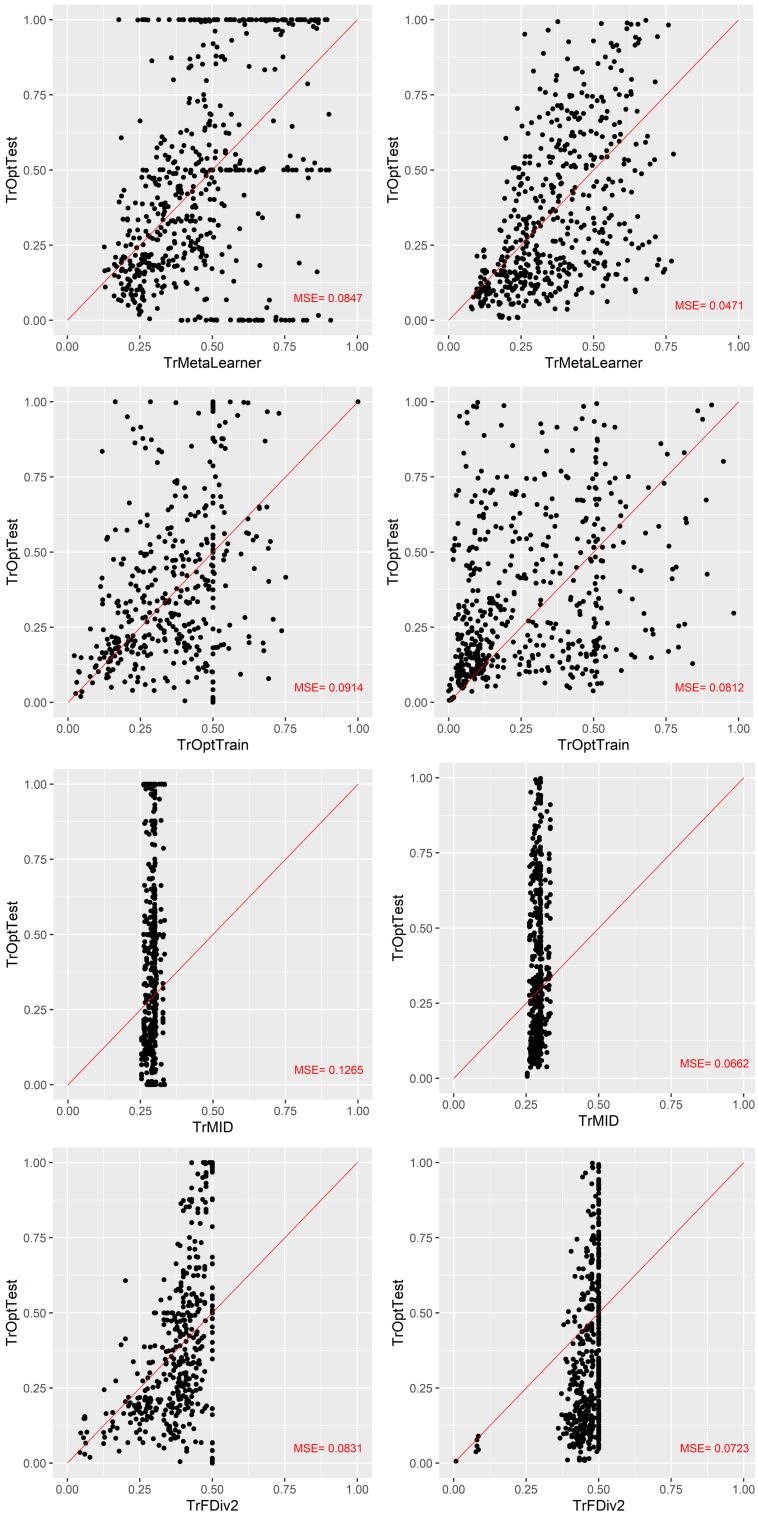
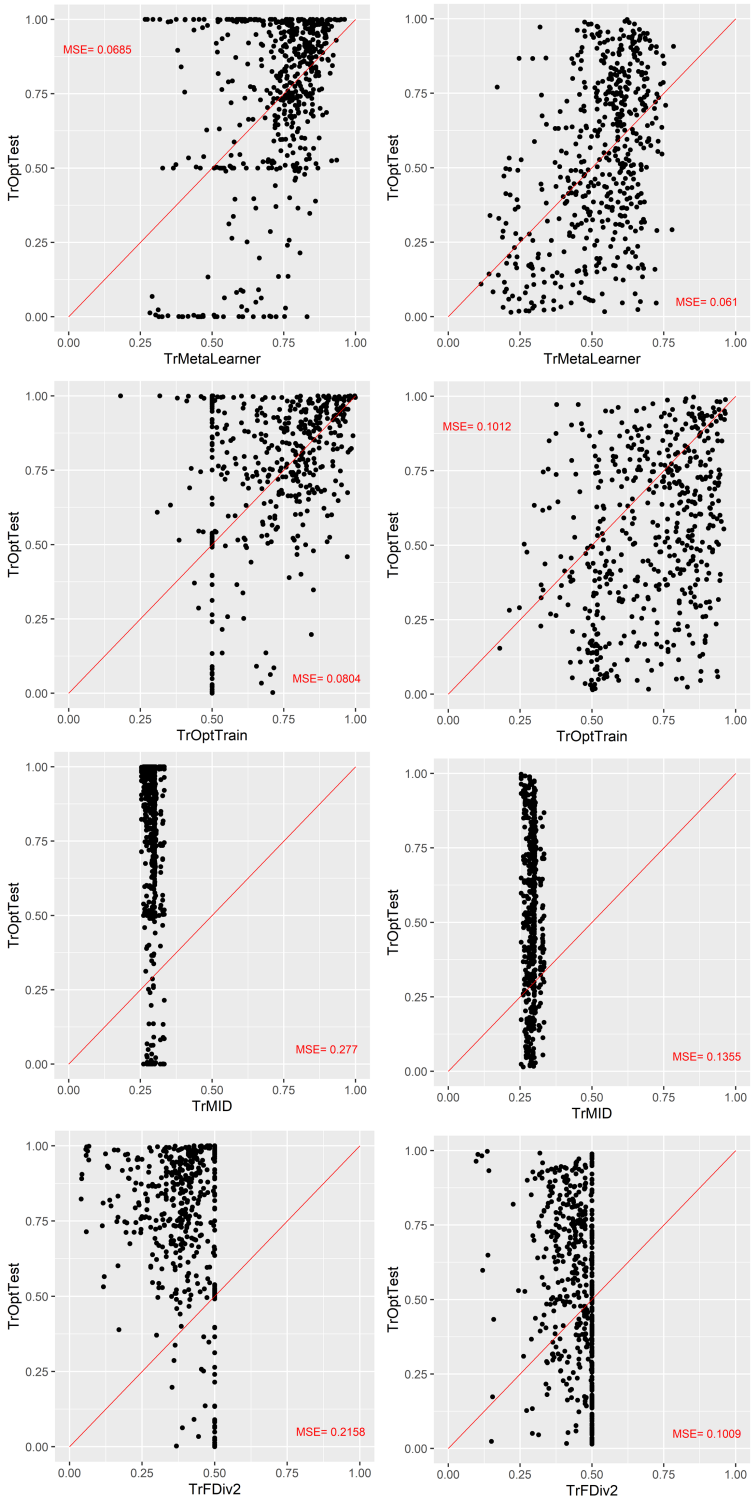


Figure 5.1: Predicted versus optimal thresholds for LR and SVM.



LR-SMOTE SVM-SMOTE

Figure 5.2: Predicted versus optimal thresholds for LR-SMOTE and SVM-SMOTE.

Table 5.5: Significance of marginal $p-values$ of the proposed meta-features.

Meta-feature	**LR**	**SVM**	**LR-SMOTE**	**SVM-SMOTE**
RatePtoN	***	***	***	***
CntPos				***
CntNegFrst	***			**
CntNegBtwn	***	***		***
ScrPosFrst	***	***		
ScrPosLst	***	***	***	***
ScrSmplP	***	***	***	***
AvgScrPtoN	***	***	***	***
AvgScrNtoP	***	*	***	***
BinRate1	***	*	***	***
BinRate2	**		*	***
BinRate3	***	***	***	
BinRate4	**	***	*	
BinRate5		*	**	
Gini	***	***	***	***
TrOptTrain	***	***	***	***
Fmax	***	***	***	***

Note: $*p \leq 0.05$, $**p \leq 0.01$, $***p \leq 0.001$.

The significance of the proposed meta-features for the threshold predictor is examined through a set of statistical tests. Particularly, 17 linear regression models are built by considering the optimal test threshold (TrOptTest) as the dependent variable and each of the features as the predictor. The marginal significance level of each feature suggested for the meta-learner is represented in Table 5.5 in terms of $p \quad value$ levels. It can be stated that most of the features are significantly related to the target threshold for all settings. Moreover, there is no predictor which is insignificant in all four scenarios.

The importance of the meta-features in Random Forest (RF) threshold predictor models is computed and illustrated in Figure 5.3 for LR, SVM, LR-SMOTE, and

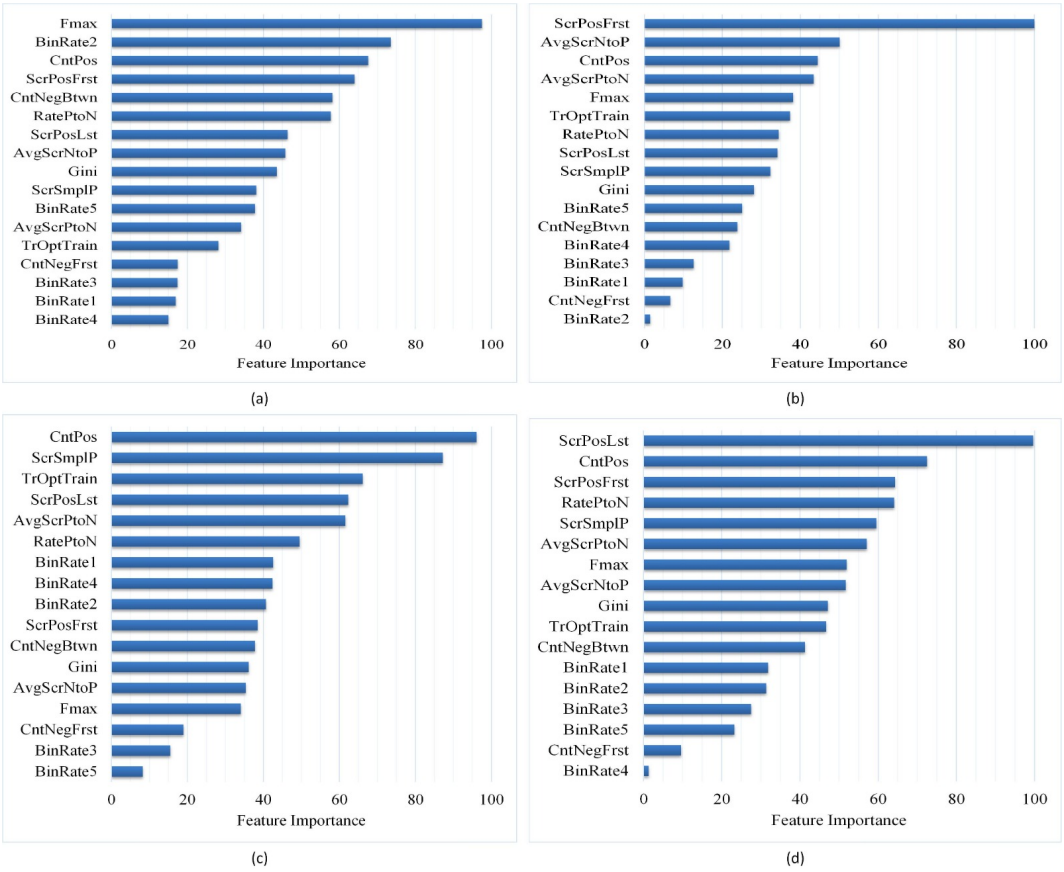


Figure 5.3: Feature importance in RF threshold predictor models for (a) LR, (b) SVM, (c) LR-SMOTE and (d) SVM-SMOTE.

SVM-SMOTE. Feature importance in RF is a measure of the decrease in node impurities in terms of Gini index from splitting on the feature. The total value is averaged across all trees in the forest. As inferred from the figure, the comprehensive set of features proposed for the meta-learner efficiently captures the inherent characteristics of the classification scores required for threshold prediction in different settings. However, the ranking of the variables from the most to the least important is not identical among the four arrangements. The only common feature among the top three features is CntPos. The first-ranked features for LR, SVM, LR-SMOTE, and SVM-SMOTE are Fmax, ScrPosFrst, CntPos, and ScrPosLst respectively. The BinRate features are among the least important ones.

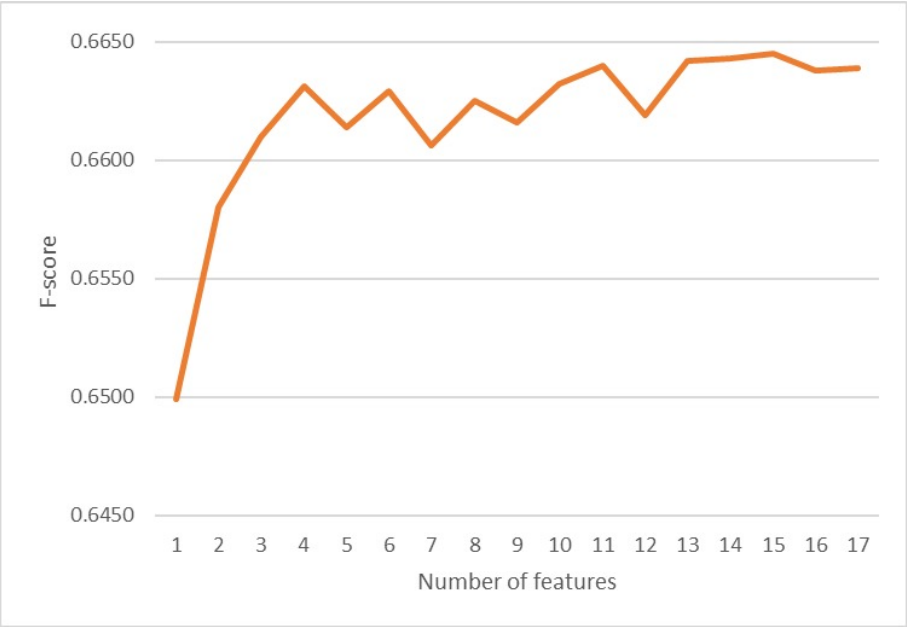


Figure 5.4: The $F-score$ values achieved using different feature sets for SVM.

In order to evaluate the effectiveness of different subsets of features, the experimental results are reported for SVM and LR for different feature subsets. More specifically, starting from the most important feature, the sorted features are added one by one to the set and, the $F-score$ values obtained are recorded. Figures 5.5 and 5.5 depict the $F-scores$ obtained. As can be seen in the figures, the features proposed form a reasonable set for the task under concern. As a matter of fact, computing a better-fitting subset is not considered to be promising for achieving further performance gain.

The relative performance of the methods is also compared on subsets of the datasets. Fifty two datasets are firstly sorted according to their imbalance ratios. Then, they are split into four equal subsets. The average $F-scores$ are reported for each group in Figures 5.6, 5.7, 5.8 and 5.9. The scores reported in Figure 5.6 represent the average $F-scores$ obtained for the most imbalanced set of thirteen datasets whereas Figure 5.9 corresponds to those having the smallest imbalances. It can be seen in the figures that the proposed approach is superior to its competitors in all subsets, but particularly

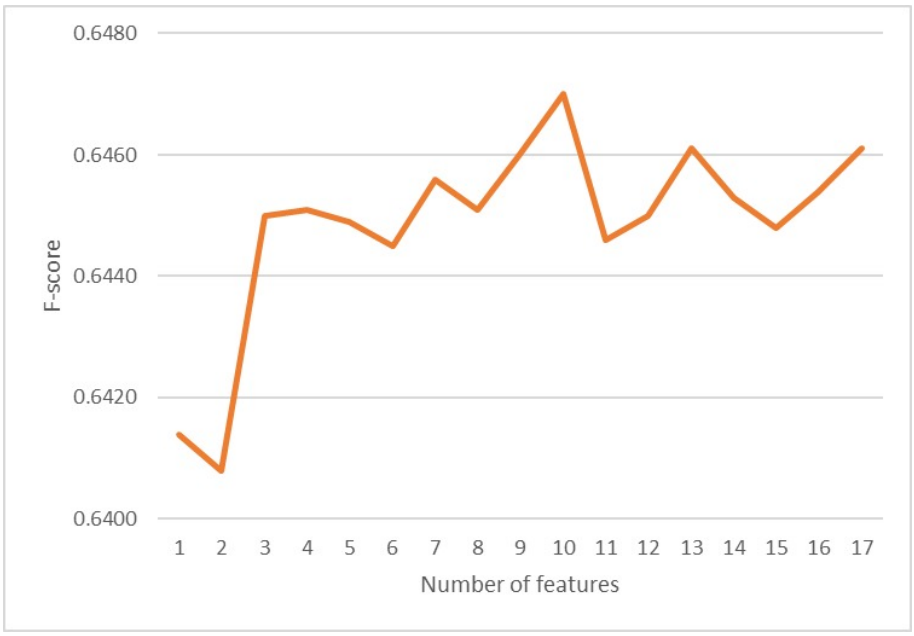


Figure 5.5: The $F-score$ values achieved using different feature sets for LR.

when the imbalance ratio is not small.

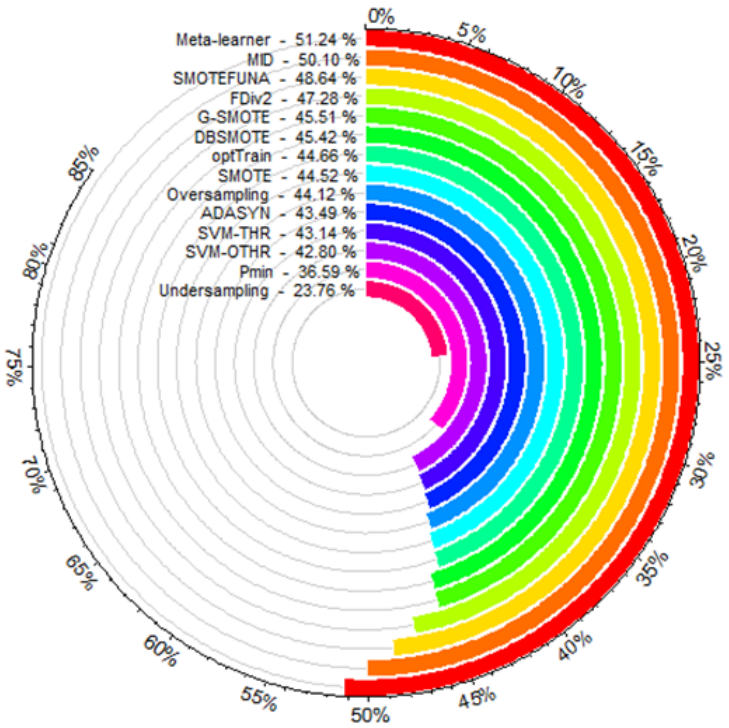


Figure 5.6: The average $F-scores$ obtained for the most imbalanced 13 datasets.

Nemenyi test is conducted to measure the significance of the improvements achieved.

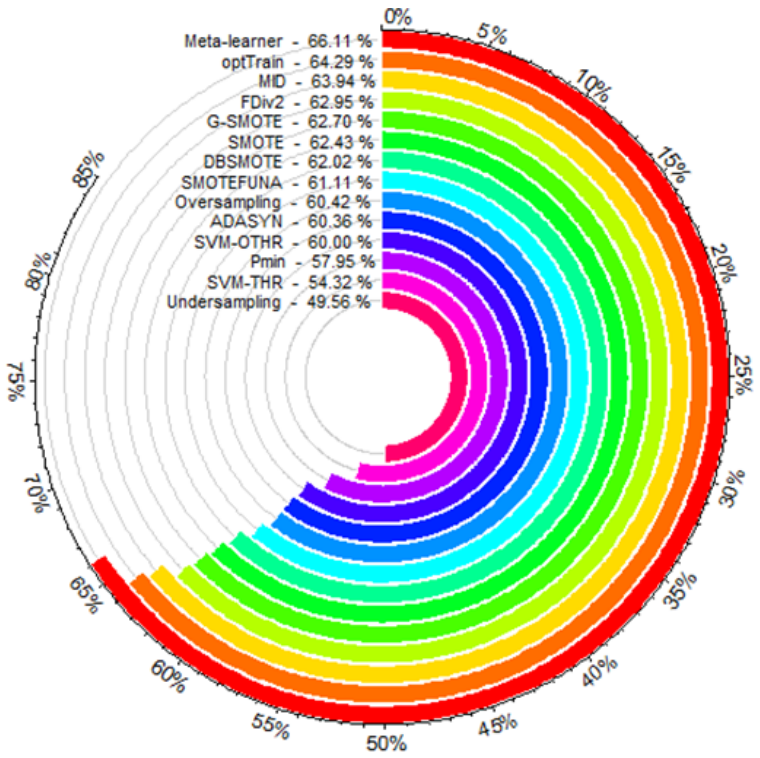


Figure 5.7: The average $F-scores$ obtained for the second group of imbalanced 13 datasets.

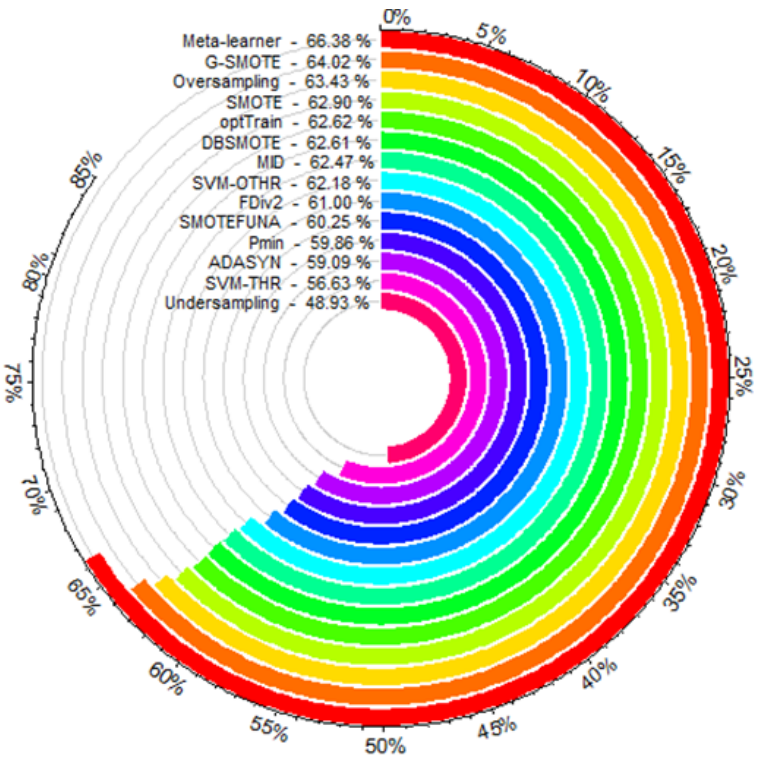


Figure 5.8: The average $F-scores$ obtained for the third group of 13 datasets.

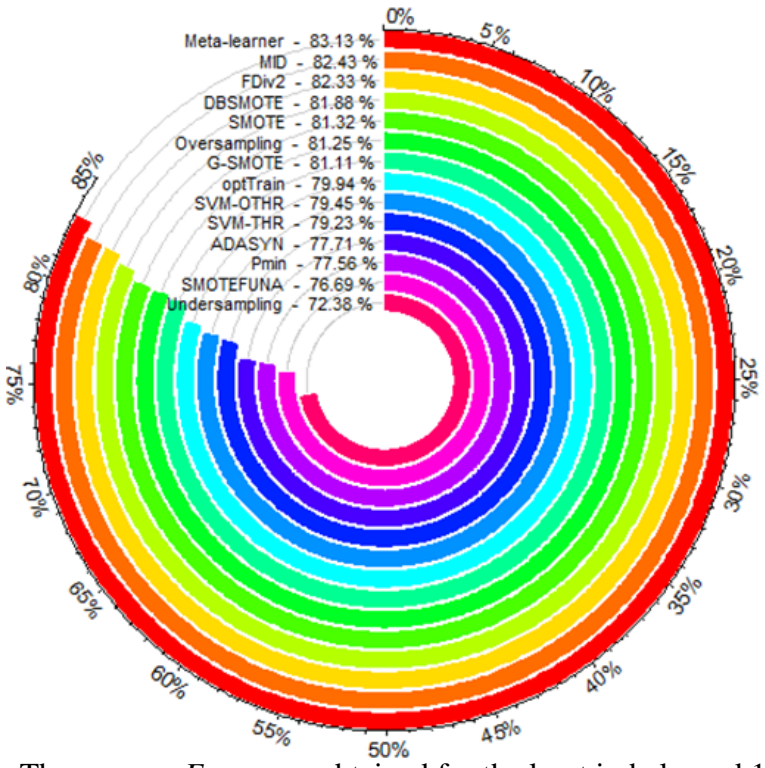


Figure 5.9: The average $F-scores$ obtained for the least imbalanced 13 datasets.

The critical distance (CD) diagram is presented in Figure 5.10. As can be seen in the figure, the meta-learner-based threshold prediction technique performs significantly better than all competing schemes except for MID.

5.3 Experiments on Evaluating MaMiPot

In this approach, since the risk of distorting the boundary is less when compared to balancing, three nonlinear classifiers are employed in the simulations. Taking into account the superior performance of SVM compared to LR in thresholding experiments, other than SVM, we used two other nonlinear classifiers namely, naive Bayes (NB) and decision tree (J48) which are also widely employed for imbalanced datasets [102]. The default settings are utilized for J48, i.e. confidence factor = 0.25 and minimum instances per leaf is 2. For each dataset, the cross-validation experiments are repeated 20 times and the average scores are reported.

The performance of the proposed method is compared with SMOTE and nineteen

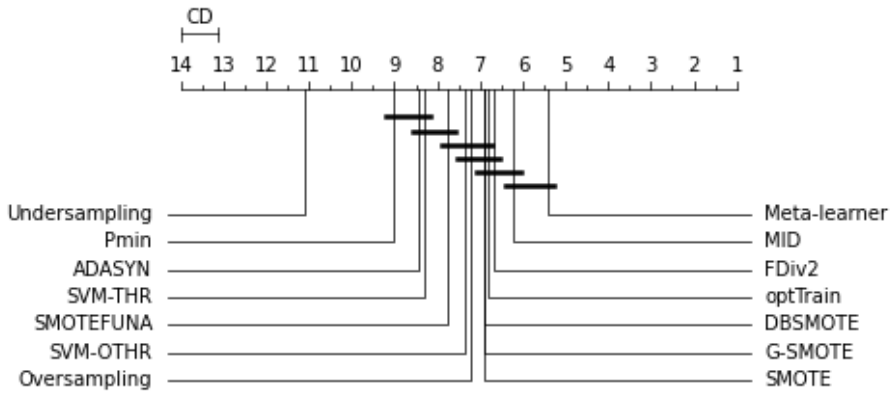


Figure 5.10: Comparative evaluation of different schemes using Nemenyi test where $\alpha = 0.05$.

extensions namely, ADASYN [82], Borderline-SMOTE1 [94], DBSMOTE [73], GSMOTE [77], SMOTEFUNA [72], MWMOTE [68], ANS [109], CCR [96], CURE-SMOTE [90], kmeans-SMOTE [93], SMOTE-D [88], MDO [91], MCT [87], SMOTE-ENN [95], SMOTE-TomekLinks [95], Polynom-fit-SMOTE [86], SMOTE-IPF [84], TRIM-SMOTE [85] and Cluster-SMOTE [89].

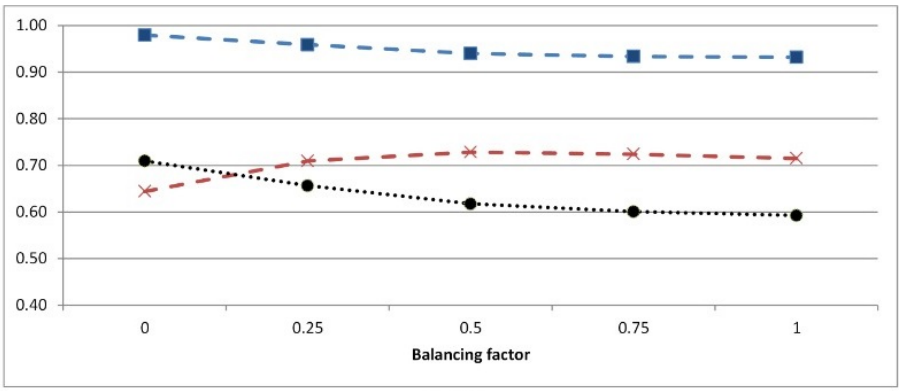
In the first set of experiments, MaMiPot is run using five balancing factors, $\beta \in \{0, 0.25, 0.50, 0.75, 1\}$. As mentioned above, α_P and α_N which are in the interval $(0, 1)$ represent the percentage of repositioning on the lines connecting the minority and majority instances to the centroids of the corresponding classes. To reposition the minority instances in small steps, we set $\alpha_P = 0.1$. On the other hand, we set $\alpha_N = 0.9$ to allow larger displacements for the majority class. A large value of α_N is expected to allow MaMiPot terminate in small number of iterations. Since the minority class is rare in many datasets, we set $\gamma = 1$. When $\beta = 0$, only MaMiPot is applied. For $\beta > 0$, SMOTE is applied after MaMiPot where the number of synthetic samples is determined by β. The results obtained are presented in Figures 5.11 and 5.12. Figure 5.11 depicts the specificity, recall and precision scores. As expected,

increasing the minority samples by applying SMOTE results in decreased precision but increased recall, when compared to using the original training set. Due to reducing the bias on the majority class using oversampling, specificity which denotes the performance on the majority samples decreases as β increases.

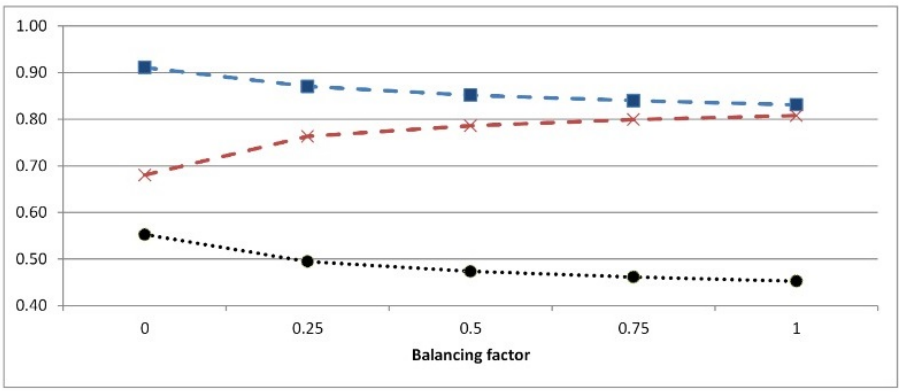
Figure 5.12 presents the performance scores obtained for three trade-off metrics (F-score, G-mean and AUC) and several balancing factors. It can be seen in the figure that, applying oversampling after MaMiPot degrades the average F-score for NB. On the other hand, G-mean improves for all three classifiers when $\beta = 0.25$. Similarly, a notable improvement is observed for AUC in the case of J48. For $\beta > 0.25$, there is not any notable improvement for SVM and NB. When all metrics are considered, it can be argued that $\beta = 0.25$ is a reasonable setting.

The average F-score, G-mean and AUC scores obtained are presented in Table 5.6. The results obtained for both $\beta = 0$ and oversampling using $\beta = 0.25$ are reported. For each classifier and metric, the highest score is presented in boldface and the second highest score is underlined. The table shows that, using $\beta = 0.25$ our method achieves the highest average scores for all three metrics for SVM. Similarly, it provides better scores than all references for F-score and G-mean using NB. Another observation is that, for all three performance metrics, the scores obtained for SVM are superior to those obtained for both NB and J48. For G-mean, the importance of applying oversampling after repositioning by MaMiPot is evident from the scores obtained using all classifiers. On the other hand, when the average performance over all folds is considered, MaMiPot is not among the top-ranked systems for J48.

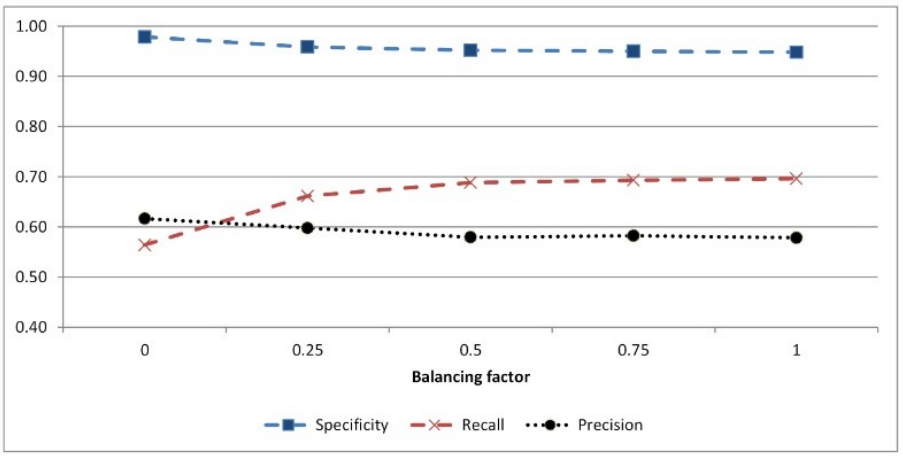
For a deeper evaluation of MaMiPot for different β, the performance scores are



(SVM)

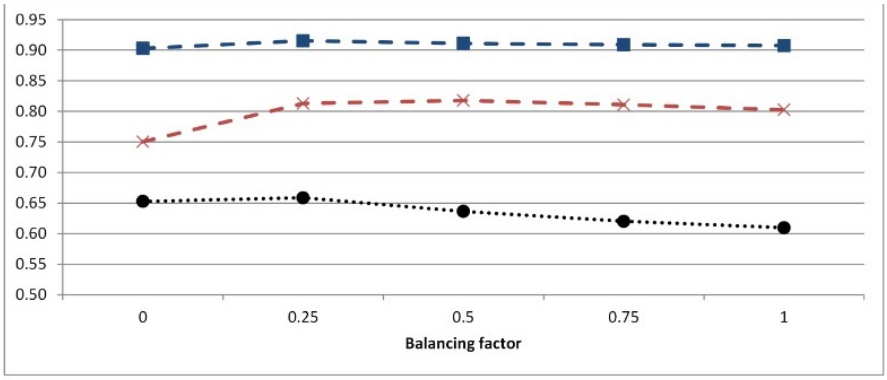


(NB)

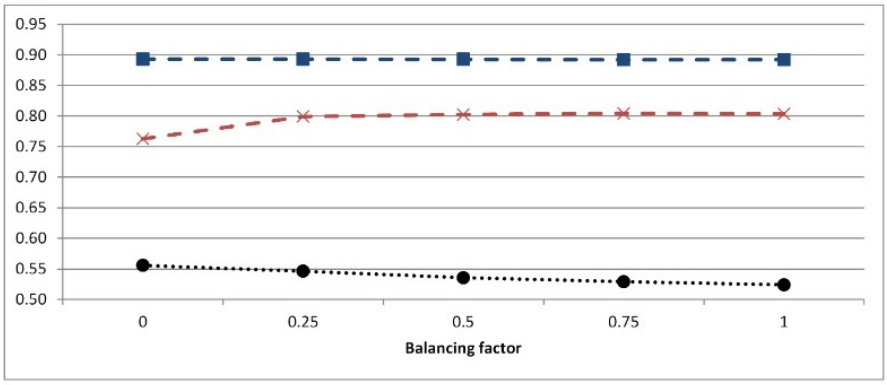


(J48)

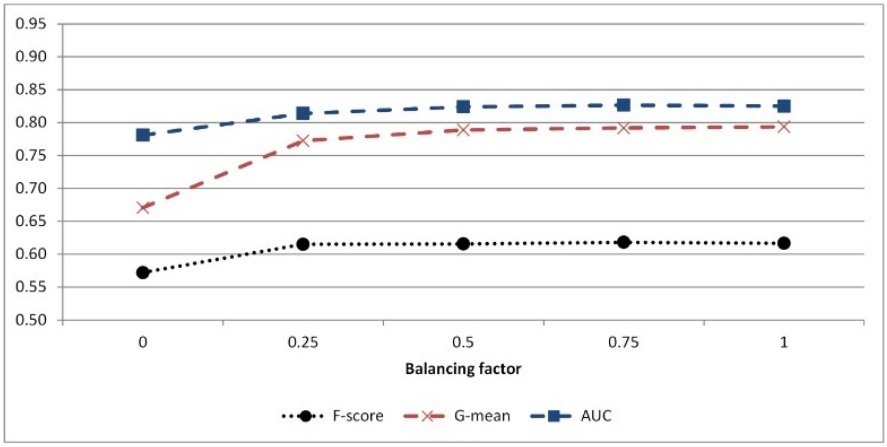
Figure 5.11: Average sensitivity (recall), specificity and precision scores obtained by MaMiPot ($\beta = 0$) and oversampling after MaMiPot using different balancing factor values, $\beta > 0$. Each row corresponds to a different classifier.



(SVM)



(NB)



(J48)

Figure 5.12: Average F-score, G-mean and AUC values obtained by MaMiPot ($\beta = 0$) and oversampling after MaMiPot using different balancing factor values, $\beta > 0$. Each row corresponds to a different classifier.

Table 5.6: The average performance scores obtained using SVM, NB and J48.

	SVM			NB			J48		
	F-score	*G*-mean	*AUC*	*F*-score	*G*-mean	*AUC*	*F*-score	*G*-mean	*AUC*
SMOTE	0.6239	0.7753	0.9080	0.5309	<u>0.7928</u>	<u>0.8946</u>	0.6152	0.7939	0.8251
ADASYN	0.5965	0.7580	0.9038	0.5058	0.7839	0.8907	0.6067	0.7897	0.8211
Borderline -SMOTE1	0.6292	0.7598	0.9084	0.5374	0.7873	0.8919	**0.6228**	0.7816	0.8192
DBSMOTE	0.6234	0.7581	0.9071	0.5396	0.7764	0.8815	0.6165	0.7674	0.8246
GSMOTE	0.6288	<u>0.8017</u>	<u>0.9127</u>	0.5200	0.7891	**0.9021**	0.6108	<u>0.8120</u>	**0.8384**
SMOTE FUNA	0.6157	0.7402	0.9034	0.5188	0.7453	0.8901	0.6126	0.7618	0.8145
MWMOTE	0.6070	0.7587	0.9026	0.4987	0.7658	0.8689	0.5993	0.7665	0.8096
ANS	0.5948	0.6874	0.8958	0.5077	0.7017	0.8548	0.6021	0.7235	0.7971
CCR	0.5736	0.7638	0.8921	0.4281	0.5742	0.8721	0.5779	0.7019	0.7978
CURE -SMOTE	0.5925	0.7017	0.8958	0.4762	0.6899	0.8574	0.5865	0.7155	0.7957
kmeans -SMOTE	0.5666	0.6336	0.8952	0.4914	0.6790	0.8605	0.6001	0.7026	0.7933
SMOTE-D	0.6058	0.7472	0.9049	0.5078	0.7579	0.8866	0.6075	0.7760	0.8152
MDO	0.5920	0.7060	0.8957	0.4430	0.6521	0.8699	0.6072	0.7299	0.8148
MCT	0.6165	0.7691	0.9083	0.4997	0.7681	0.8710	0.6057	0.7588	0.8032
SMOTE -ENN	0.6067	0.7835	0.9061	0.5257	0.7855	0.8942	0.6059	**0.8132**	<u>0.8326</u>
SMOTE-TomekLinks	0.6228	0.7737	0.9078	0.5311	0.7924	0.8938	0.6146	0.7941	0.8244
Polynom fit-SMOTE	0.6255	0.7534	0.9082	0.5118	0.7238	0.8794	0.6079	0.7261	0.8139
SMOTE -IPF	0.6225	0.7733	0.9080	0.5299	0.7921	0.8942	<u>0.6157</u>	0.7939	0.8252
TRIM -SMOTE	0.6237	0.7288	0.9022	0.5399	0.7585	0.8805	0.6124	0.7458	0.8042
Cluster -SMOTE	0.6046	0.7428	0.9038	0.4989	0.7392	0.8692	0.6011	0.7583	0.8049
MaMiPot ($\beta = 0$)	<u>0.6528</u>	0.7503	0.9030	**0.5558**	0.7626	0.8928	0.5720	0.6705	0.7809
MaMiPot ($\beta = 0.25$)	**0.6587**	**0.8129**	**0.9153**	<u>0.5462</u>	**0.7991**	0.8929	0.6151	0.7722	0.8138

Table 5.7: The rankings of MaMiPot for different β values. The averages of all nine ranks are presented in the last column.

	SVM			NB			J48			Avg.
	F-score	*G*-mean	*AUC*	*F*-score	*G*-mean	*AUC*	*F*-score	*G*-mean	*AUC*	Rank
MaMiPot ($\beta = 0.25$)	1	2	1	2	1	2	1	4	4	**2.00**
MaMiPot ($\beta = 0.50$)	3	1	3	3	2	3	3	3	3	2.67
MaMiPot ($\beta = 0.75$)	4	3	4	4	3	4	2	2	1	3.00
MaMiPot ($\beta = 0.00$)	2	5	2	1	5	1	5	5	5	3.44
MaMiPot ($\beta = 1.00$)	5	4	5	5	4	5	4	1	2	3.89

compared in terms of the average ranks over all folds for different values. In particular, a rank score is assigned for each β value and each fold. Then, the sums of these ranks over all folds are calculated for each β. After sorting in ascending order, the overall ranks are recorded. Table 5.7 presents the overall ranks corresponding to each classifier and performance metric pair. The last column presents the row-wise averages. $\beta = 0.25$ is found to be the top-ranked when the ranking performance is evaluated for all metrics and classifiers. It can be seen in the table that, when *G*-mean and *AUC* are considered, J48 benefits more from higher β values when compared to SVM and NB.

Taking into account the ranking scores presented in Table 5.7, we set $\beta = 0.25$ in the following context. Table 5.8 presents the overall rank of each classifier and performance metric pair. The last column presents the row-wise averages. It can be seen in Table 5.8 that, MaMiPot using $\beta = 0.25$ is the top-ranked technique when the ranking performance is evaluated for all metrics and classifiers. It should be noted that, when evaluated using *F*-score, MaMiPot is top-ranked for SVM and second ranked for both NB and J48. The ranks achieved using J48 are low for *G*-mean and *AUC*. In a recent study, the performances of seven classifiers are evaluated on datasets

Table 5.8: The rankings of different schemes according to their fold-based performance scores for SVM, NB and J48. The averages of all nine ranks are presented in the last column.

	SVM			NB			J48			Avg.
	F-score	*G*-mean	*AUC*	*F*-score	*G*-mean	*AUC*	*F*-score	*G*-mean	*AUC*	Rank
MaMiPot	1	1	2	2	1	7	2	10	13	**4.33**
SMOTE-IPF	6	6	7	7	3	5	4	3	4	5.00
Borderline-SMOTE1	4	10	3	4	6	4	1	7	7	5.11
SMOTE	9	7	4	8	4	2	6	5	3	5.33
SMOTE-TomekLinks	5	5	8	6	2	6	8	4	5	5.44
GSMOTE	8	2	1	14	7	1	16	2	1	5.78
SMOTE-ENN	16	3	11	9	5	3	20	1	2	7.78
TRIM-SMOTE	2	14	10	1	8	8	3	14	16	8.44
DBSMOTE	10	13	9	3	11	17	9	11	6	9.89
Polynom-fit-SMOTE	3	12	6	10	16	19	10	17	11	11.56
MCT	7	4	5	16	10	10	15	16	21	11.56
SMOTE-D	13	16	16	13	13	9	14	9	10	12.56
SMOTEFUNA	11	17	13	12	15	16	5	13	12	12.67
Cluster-SMOTE	12	15	15	15	14	12	13	12	15	13.67
MDO	17	11	12	21	18	14	7	15	9	13.78
MWMOTE	15	9	18	18	12	13	17	8	14	13.78
ADASYN	19	18	19	19	9	11	18	6	8	14.11
ANS	14	19	17	5	17	18	12	18	17	15.22
kmeans-SMOTE	20	21	14	11	19	15	11	20	20	16.78
CCR	21	8	20	20	21	20	21	21	19	19.00
CURE-SMOTE	18	20	21	17	20	21	19	19	18	19.22

Table 5.9: The average standard deviations over all folds for MaMiPot.

	F-score	G-mean	AUC
SVM	0.0326	0.0234	0.1005
NB	0.0372	0.0316	0.1128
J48	0.2560	0.1929	0.1404

that are grouped according to their characteristics [16]. Considering the set of 5 neighboring samples, rare and outlying instances are defined to have one or zero minority instances in their sets of 5 neighbors, respectively. A sample is labeled as safe if at most one of its neighbors is from the majority class. It is shown that SVM performs better than J48 on datasets which include safe and borderline instances [16]. It can be argued that, by repositioning the majority samples away from the minority class, training sets are transformed into safer forms, leading to better performance for SVM, when compared to J48.

It is well known that decision trees are unstable classifiers whereas SVM and NB are stable [110]. Training unstable classifiers like decision trees is challenging when the training set size is small because a small perturbation in the training set may result in a highly different model, leading to substantial differences in the performance scores obtained on unseen test sets [111, 112]. As a matter of fact, the standard deviations in performance scores obtained from different folds are expected to be larger for unstable classifiers. In fact, variations in the predicted labels due to small perturbations on the training sets is effectively used in constructing classifier ensembles [113, 114]. In this approach, the predictions of unstable learners such as decision trees are combined to obtain more accurate decisions [115]. Table 5.9 presents the average standard deviations for 20 repetitions over all folds for MaMiPot. It can be seen that the standard deviations are much larger for J48. For a better

Table 5.10: The average performance scores obtained using SVM, NB and J48 using 5-fold cross validation.

	SVM			NB			J48		
	F-score	*G*-mean	*AUC*	*F*-score	*G*-mean	*AUC*	*F*-score	*G*-mean	*AUC*
SMOTE	0.6477	0.8049	0.9136	0.5390	**0.7994**	0.9044	0.6223	0.7939	0.8326
ADASYN	0.6150	0.7845	0.9088	0.5053	0.7836	0.9011	0.6099	0.7852	0.8276
Borderline -SMOTE1	0.6605	0.8005	0.9168	0.5391	0.7925	0.8991	0.6354	0.7876	0.8323
DBSMOTE	0.6550	0.7925	0.9144	0.5499	0.7793	0.8903	0.6327	0.7696	0.8378
GSMOTE	0.6538	0.8332	0.9199	0.5308	0.7972	**0.9138**	0.6306	0.8165	**0.8532**
SMOTE FUNA	0.6459	0.7616	0.9102	0.5305	0.7460	0.8984	0.6419	0.7744	0.8392
MWMOTE	0.6210	0.7806	0.9147	0.5265	0.7912	0.8934	0.6186	0.7803	0.8266
ANS	0.6492	0.7427	0.9119	0.5547	0.7735	0.8836	0.6358	0.7503	0.8117
CCR	0.5752	0.7473	0.8804	0.4394	0.5580	0.8785	0.6236	0.7358	0.8228
CURE -SMOTE	0.6330	0.7328	0.9031	0.4929	0.6979	0.8622	0.6207	0.7221	0.8115
kmeans -SMOTE	0.6309	0.6855	0.9103	0.5240	0.7098	0.8691	0.6460	0.7400	0.8220
SMOTE_D	0.6371	0.7892	0.9114	0.5144	0.7570	0.8996	0.6288	0.7873	0.8290
MDO	0.6113	0.7245	0.8972	0.4575	0.6636	0.8767	0.6269	0.7388	0.8350
MCT	0.6376	0.8003	0.9161	0.5237	0.7819	0.8904	0.6319	0.7711	0.8186
SMOTE ENN	0.6300	0.8130	0.9141	0.5353	0.7864	0.9024	0.6278	**0.8234**	0.8518
SMOTE -TomekLinks	0.6476	0.8025	0.9134	0.5336	0.7942	0.9058	0.6349	0.8019	0.8367
polynom -fit SMOTE	0.6578	0.7857	0.9142	0.5300	0.7421	0.8877	0.6213	0.7190	0.8254
SMOTE -IPF	0.6462	0.8017	0.9138	0.5381	0.7983	0.9076	0.6203	0.7910	0.8355
TRIM -SMOTE	0.6656	0.7652	0.9166	0.5449	0.7815	0.8952	0.6468	0.7707	0.8267
Cluster -SMOTE	0.6313	0.7752	0.9136	0.4899	0.7384	0.8835	0.6332	0.7782	0.8244
MaMiPot ($\beta = 0.25$)	**0.6815**	**0.8358**	**0.9236**	0.5616	0.7927	0.8992	**0.6529**	0.7948	0.8327

Table 5.11: The rankings of different schemes according to their fold-based performance scores for SVM, NB and J48 using 5-fold cross validation. The averages of all nine ranks are presented in the last column.

	SVM			NB			J48			Avg. Rank
	F-score	*G*-mean	*AUC*	*F*-score	*G*-mean	*AUC*	*F*-score	*G*-mean	*AUC*	
MaMiPot	1	1	2	2	3	7	2	3	10	**3.44**
TRIM_SMOTE	2	8	3	3	1	6	1	4	5	3.67
ANS	4	15	11	1	2	10	4	6	11	7.11
GSMOTE	13	2	1	16	11	1	16	2	2	7.11
SMOTE_TomekLinks	8	3	9	10	7	4	14	5	7	7.44
Borderline_SMOTE1	6	7	5	11	9	8	6	11	9	8.00
DBSMOTE	5	11	10	4	5	18	9	9	3	8.22
SMOTE_IPF	11	6	7	8	4	2	17	8	14	8.56
SMOTEFUNA	7	16	14	5	8	15	5	7	4	9.00
SMOTE	9	5	8	9	6	3	20	14	12	9.56
SMOTE_ENN	19	10	13	13	10	5	19	1	1	10.11
polynom_fit_SMOTE	3	9	4	7	12	19	13	21	8	10.67
kmeans_SMOTE	10	21	12	6	15	13	3	17	13	12.22
MCT	14	4	6	14	14	12	11	16	21	12.44
SMOTE_D	15	13	18	15	16	11	12	13	15	14.22
MWMOTE	17	12	17	17	13	9	18	10	18	14.56
Cluster_SMOTE	16	14	15	19	17	17	10	12	17	15.22
MDO	18	17	16	18	20	16	7	19	6	15.22
CURE_SMOTE	12	19	19	12	19	21	8	18	20	16.44
ADASYN	20	18	20	20	18	14	21	15	16	18.00
CCR	21	20	21	21	21	20	15	20	19	19.78

evaluation of the relative performances of the schemes considered, 5-fold cross-validation is also performed. In this approach, 80% of the data is employed during training instead of 50%. The average scores and the corresponding ranks are presented in Tables 5.10 and 5.11, respectively. Comparing the results from Tables 5.8 and 5.11, for J48, the average ranks of MaMiPot are improved from 10 to 3 for G-mean and 13 to 10 for AUC. For stable classifiers SVM and NB, only the rank for G-mean is degraded from 1 to 3 in the case of NB. Taking into account the size of datasets employed and the observations listed above, we argue that the inferior rankings achieved using J48 can be attributed to its unstable behavior on small datasets.

The KEEL repository includes datasets having a wide range of imbalance ratios. Similarly, the numbers of positive samples and dimensionalities of the feature vectors are different. In order to investigate the relative performance of different techniques for datasets having different characteristics, the average ranks of different KEEL subsets are computed. The results are presented in Table 5.12. The scores reported for $IR \geq 10$ are obtained using the datasets for which the imbalance ratio is greater than or equal to ten. Similarly, the column labeled as $P > 30$ is for the list of datasets where the number of positive samples is above 30. In the last two columns, the datasets are partitioned according to the number of available features, d. MaMiPot is top-ranked for highly imbalanced datasets. Similarly, it is the best-performing scheme when the number of positive samples is small. This is in line with the main target that was set as alleviating the imbalance problem without altering the distribution of the minority class.

Table 5.12: The ranks on different subsets of KEEL datasets. The numbers in parentheses indicate the order of top 5 schemes.

	$IR \geq 10$	$IR < 10$	$P > 30$	$P \leq 30$	$d \geq 9$	$d < 9$
MaMiPot	$3.00^{(1)}$	9.78	$6.56^{(3)}$	$3.33^{(1)}$	7.44	$4.00^{(1)}$
SMOTE-IPF	$6.44^{(5)}$	$5.33^{(1)}$	$5.00^{(1)}$	6.44	$3.89^{(2)}$	$7.89^{(5)}$
Borderline-SMOTE1	$4.11^{(4)}$	9.56	$7.00^{(5)}$	$4.89^{(2)}$	7.22	$5.67^{(3)}$
SMOTE	$4.11^{(3)}$	$8.89^{(5)}$	7.22	$5.22^{(4)}$	$6.22^{(5)}$	$6.00^{(4)}$
SMOTE-TomekLinks	7.22	$5.44^{(2)}$	$5.33^{(2)}$	6.67	$3.67^{(1)}$	8.33
GSMOTE	$3.22^{(2)}$	9.33	7.89	$5.00^{(3)}$	$5.67^{(4)}$	$5.33^{(2)}$
SMOTE-ENN	8.22	$7.33^{(4)}$	7.44	9.11	$5.11^{(3)}$	9.67
TRIM-SMOTE	11.00	$6.78^{(3)}$	$6.89^{(4)}$	8.33	9.33	9.33
DBSMOTE	6.67	13.33	12.67	$6.33^{(5)}$	10.22	10.11
Polynom-fit-SMOTE	12.89	10.22	12.44	10.44	10.67	12.33
MCT	14.22	9.22	10.78	12.56	10.56	12.67
SMOTE-D	12.33	12.33	11.11	13.78	8.33	15.22
SMOTEFUNA	9.33	15.78	15.22	10.33	14.33	12.67
Cluster-SMOTE	13.89	11.67	11.78	15.00	14.00	13.22
MWMOTE	15.11	10.67	12.00	15.44	12.22	13.44
MDO	16.56	11.11	12.00	14.78	20.00	8.00
ADASYN	8.11	16.11	14.33	13.22	10.67	14.56
ANS	18.33	9.89	11.78	16.44	18.89	11.22
kmeans-SMOTE	18.00	13.67	15.56	16.67	18.78	14.11
CCR	19.78	16.33	19.56	17.56	17.22	17.89
CURE-SMOTE	18.44	18.22	18.44	19.44	16.56	19.33

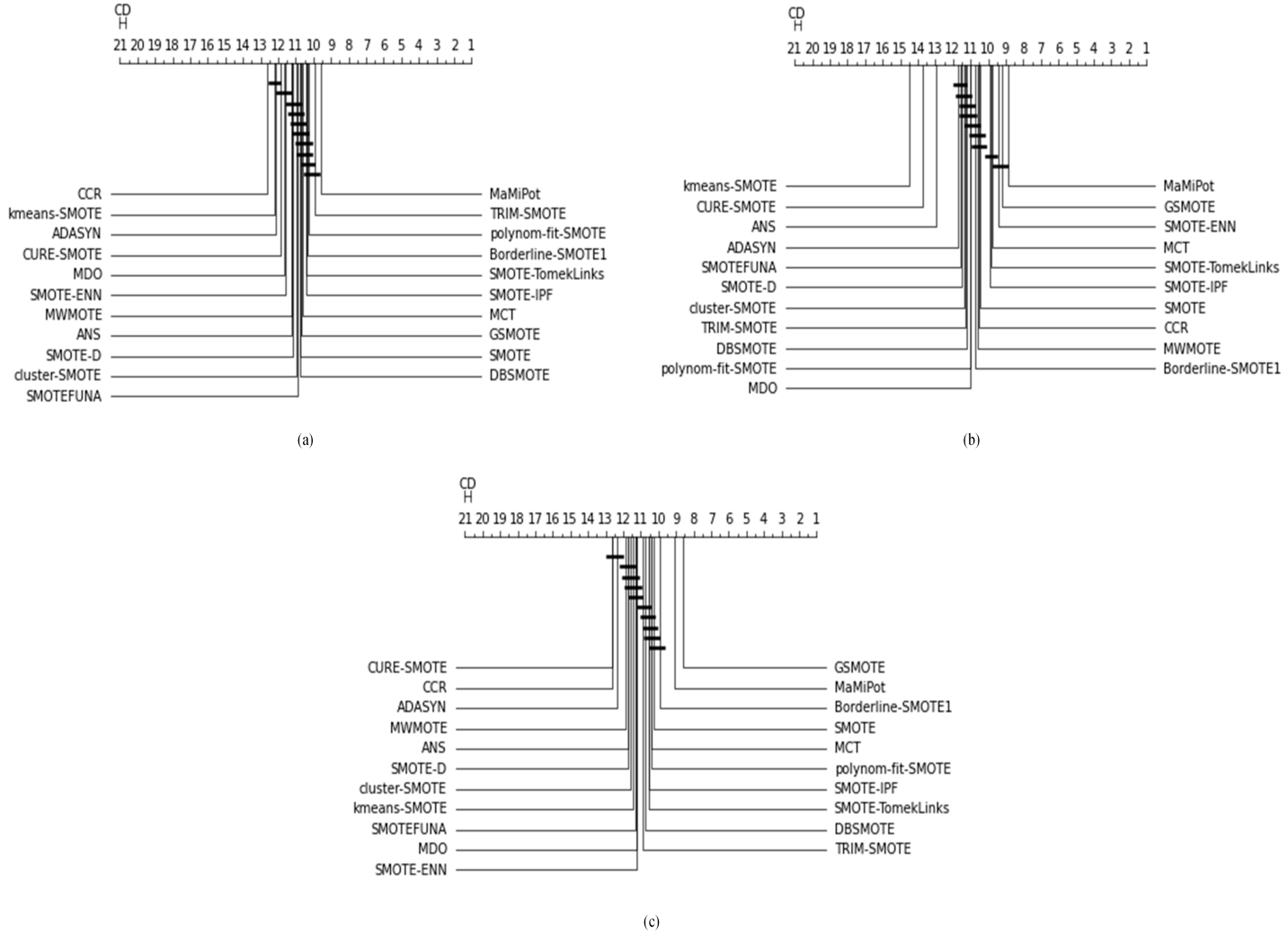


Figure 5.13: Statistical test results for SVM: (a) Using F-scores, (b) Using G-mean scores (c) Using AUC scores.

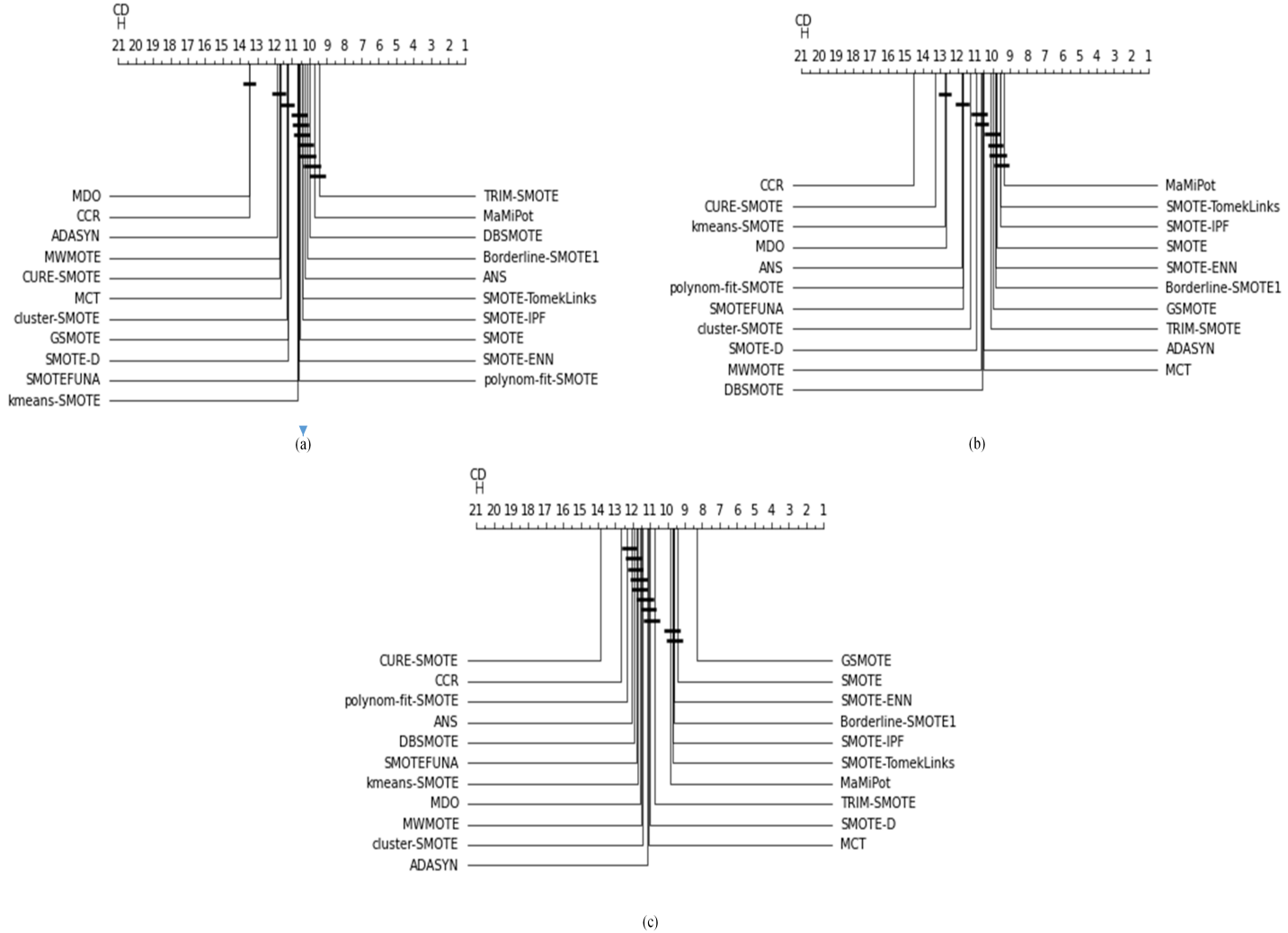


Figure 5.14: Statistical test results for NB: (a) Using F-scores, (b) Using G-mean scores (c) Using AUC scores.

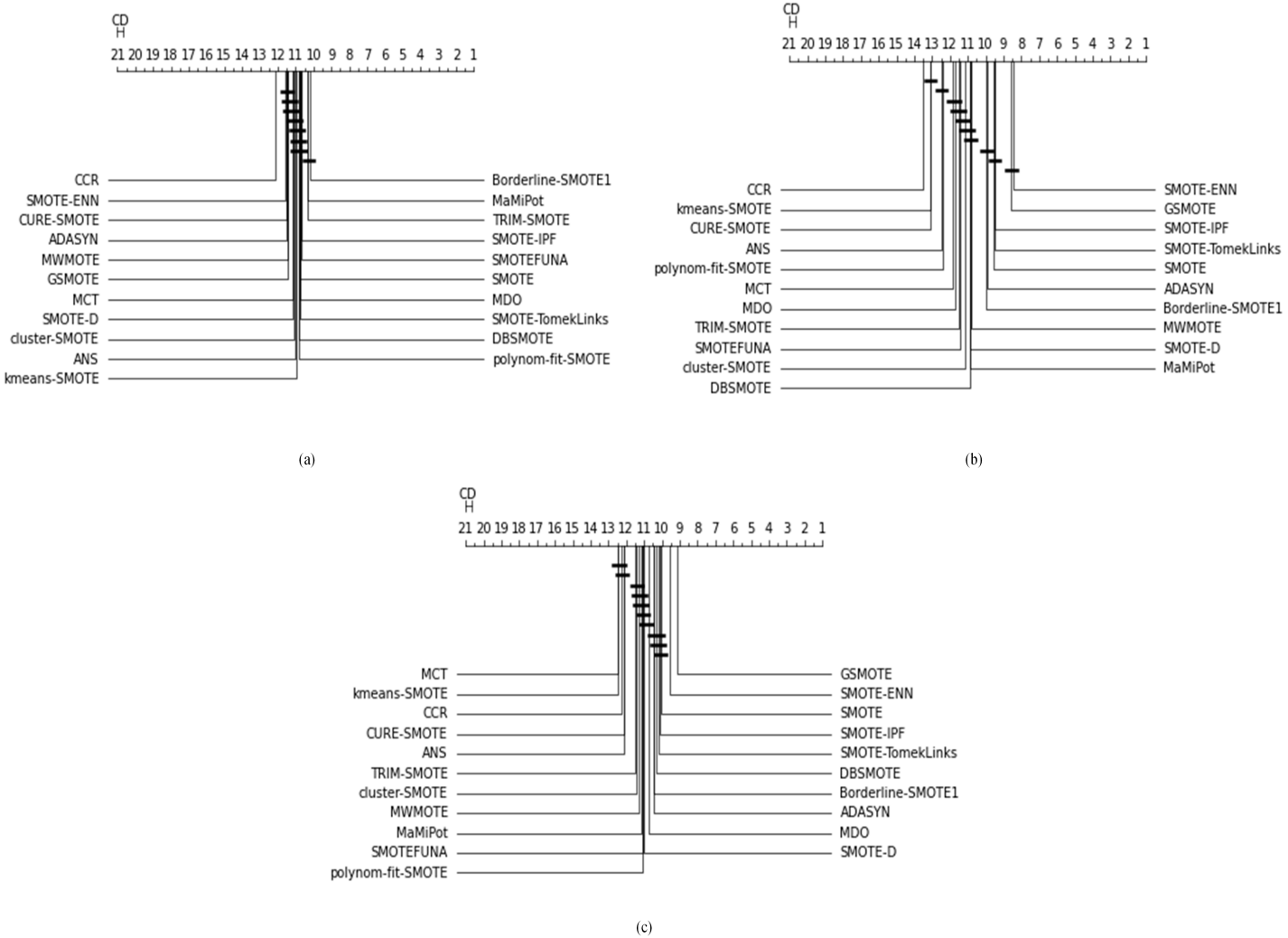


Figure 5.15: Statistical test results for J48: (a) Using F-scores, (b) Using G-mean scores (c) Using AUC scores.

Table 5.13: The ties of the method providing the highest average rank over all folds for each classifier and performance metric pair.

Classifier	Perf. Measure	Winner	Tying Methods and the p-values
SVM	F-score	MaMiPot	-
SVM	G-mean	MaMiPot	-
SVM	AUC	GSMOTE	-
NB	F-score	TRIM-SMOTE	MaMiPot (0.062)
NB	G-mean	MaMiPot	SMOTE-IPF (0.654), SMOTE-TomekLinks (0.717)
NB	AUC	GSMOTE	-
J48	F-score	Borderline-SMOTE1	MaMiPot (0.900), TRIM-SMOTE (0.900)
J48	G-mean	SMOTE-ENN	GSMOTE (0.900)
J48	AUC	GSMOTE	-

As another comparison of different approaches, Nemenyi test [116] is performed as shown in Figures 5.13, 5.14 and 5.15, respectively for SVM, NB and J48. The black lines connect the methods for which the difference of mean ranks is smaller than the critical difference (CD). The test is done in a pairwise manner, for three classifiers and three different performance metrics. In part (a), F-scores are compared. Parts (b) and (c) compare G-mean and AUC scores, respectively. It can be observed that significantly better scores than most of the reference techniques are obtained for the majority of the classifier/metric pairs, especially for SVM and NB. Although MaMiPot is not among the top-ranked schemes for G-mean and AUC when J48 is used, it can be seen in Figure 5.15 that the only method that is consistently in top five for all three metrics is SMOTE-IPF. However, this technique is not ranked in top five for any metric in the case of SVM. Table 5.13 presents the ties of the method that is top-ranked according to Table 5.6 for each classifier and performance metric. Pairwise comparisons between individual systems are performed using Nemenyi post-hoc test and the pairs having p value $\geq 0\,05$ are reported. The corresponding p-values are also provided. It can

be seen that MaMiPot is tied with the winner, TRIM-SMOTE in the case of NB and F-score. Similarly, MaMiPot is tied with the winner, borderline-SMOTE1 in the case of J48 and F-score.

Chapter 6

CONCLUSIONS AND FUTURE WORK

In this thesis, the prediction of the optimal threshold in imbalanced classification including rare positive class is first addressed. The dependence of the optimal threshold on the performance metric is analytically verified. It is shown that the best-fitting threshold for $F-score$ is higher than that of $G-mean$ in imbalanced classification problems. Threshold estimation to achieve a better $F-score$ is then addressed by considering a random forest-based model that is trained using external datasets. The model employs 17 meta-features that are defined to capture the ordering of positive and negative training samples in terms of the probability scores generated by the classifiers. Experiments have shown that threshold-moving using the proposed meta-learner is more effective than both reference thresholding methods and balancing-based ensembles. Moreover, the proposed model is able to generate better thresholds than the default for balancing-based ensembles as well.

Repositioning of the instances is proposed as an alternative approach for imbalance learning is also studied. The main motivation behind the proposed approach was to effectively learn the minority decision regions without distorting the true distribution. The proposed approach is evaluated on 52 datasets, covering a wide range of imbalance ratios. The experiments conducted using three different classifiers have shown that MaMiPot achieves the best average ranking score when evaluated in terms of three different performance metrics. The relative performance of MaMiPot and reference techniques were also compared for six groups of datasets which were formed by taking

into account the imbalance ratios, number of positive samples and number of features. Among these groups, it is observed that MaMiPot is particularly effective on datasets that have a higher imbalance ratio and on those having a smaller number of positive samples.

The experiments conducted have shown that the proposed thresholding and repostioning methods trained on the original (i.e. imbalanced) data achieve better performance scores when compared with utilizing their balanced forms. These results are in line with the criticisms of SMOTE for its distortions on the underlying distributions of the datasets.

The main property of the proposed meta-learned-based thresholding approach is that it uses external datasets during development. In other words, the training samples of the target dataset are not considered in building the meta-learner. As an alternative approach, the meta-learner trained using external datasets can be further tuned by using the training set of the target dataset before threshold prediction. Several candidate methods can be considered for this purpose. For instance, some trees in the prediction models may be adapted to the test dataset. Alternatively, the trees in the random forest-based model that have large mean square error may be replaced by new trees generated using the training set of the target dataset. The potential of these approaches will be explored in our future studies.

In spite of its inferior performance when compared to the meta-learning-based thresholding scheme, MaMiPot deserves further investigation due to being comparatively simpler. In our experimental work, we considered MaMiPot as a preprocessing algorithm only for SMOTE. As a further study, the performance of

MaMiPot should also be assessed for the variants of SMOTE. As it can be seen on lines 14 and 19 of the algorithm, the misclassified samples are moved towards the centroids denoted by μ_P and μ_N. Various alternatives can be considered for this purpose. For instance, the samples may be clustered as done in some variants of the SMOTE and the samples in sFP and sFN are moved towards the nearest cluster. Alternatively, given a majority sample in sFP, a randomly selected sample in sTN can be selected. In other words, misclassified majority samples may be moved towards a randomly selected true negative instance.

MaMiPot should be evaluated by using parameters other than the imbalance ratio. For instance, the datasets can be partitioned into subsets, each of which is dominated by either safe, borderline, outlier, or rare samples. After categorizing the datasets, MaMiPot should be run on each group to further explore its strengths and weaknesses. Moreover, MaMiPot can be modified to reposition the majority samples in different ways by taking into account the types of neighboring minority samples.

REFERENCES

[1] H. He and Y. Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*, 1st ed. Wiley-IEEE Press, 2013.

[2] A. Fernández, S. García, M. Galar, R. Prati, B. Krawczyk, and F. Herrera, *Learning from Imbalanced Data Sets*. Springer, 01 2018.

[3] S. Fotouhi, S. Asadi, and M. W. Kattan, "A comprehensive data level analysis for cancer diagnosis on imbalanced data," *Journal of Biomedical Informatics*, vol. 90, p. 103089, 2019.

[4] X. Jing, F. Wu, X. Dong, and B. Xu, "An improved sda based defect prediction framework for both within-project and cross-project class-imbalance problems," *IEEE Transactions on Software Engineering*, vol. 43, no. 4, pp. 321–339, 2017.

[5] J. Fdez-Glez, D. Ruano-Ordás, F. Fdez-Riverola, J. R. Méndez, R. Pavón, and R. Laza, "Analyzing the impact of unbalanced data on web spam classification," in *Distributed Computing and Artificial Intelligence, 12th International Conference*, S. Omatu, Q. M. Malluhi, S. R. Gonzalez, G. Bocewicz, E. Bucciarelli, G. Giulioni, and F. Iqba, Eds. Cham: Springer International Publishing, 2015, pp. 243–250.

[6] T. M. Padmaja, N. Dhulipalla, P. R. Krishna, R. S. Bapi, and A. Laha, "An unbalanced data classification model using hybrid sampling technique for fraud detection," in *Pattern Recognition and Machine Intelligence*, A. Ghosh, R. K. De, and S. K. Pal, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 341–348.

[7] C. A. Bahnsen, A. Stojanovic, D. Aouada, and E. B. Ottersten, "Improving credit card fraud detection with calibrated probabilities," in *Proceedings of the 2014 SIAM International Conference on Data Mining (SDM)*, 2014, pp. 677–685.

[8] B. Zhu, B. Baesens, and S. K. L. M. vanden Broucke, "An empirical comparison of techniques for the class imbalance problem in churn prediction," *Information Sciences*, vol. 408, pp. 84–99, 2017.

[9] J. Lee and K. Park, "GAN-based imbalanced data intrusion detection system," *Personal and Ubiquitous Computing*, vol. 25, pp. 121–128, 2021.

[10] R. Alotaibi and P. Flach, "Multi-label thresholding for cost-sensitive classification," *Neurocomputing*, vol. 436, pp. 232–247, 2021.

[11] I. Pillai, G. Fumera, and F. Roli, "Threshold optimisation for multi-label classifiers," *Pattern Recognition*, vol. 46, no. 7, pp. 2055–2065, 2013.

[12] J. Ramón Quevedo, O. Luaces, and A. Bahamonde, "Multilabel classifiers with a probabilistic thresholding strategy," *Pattern Recognition*, vol. 45, no. 2, pp. 876–883, 2012.

[13] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Int Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[14] A. N. Tarekegn, M. Giacobini, and K. Michalak, "A review of methods for imbalanced multi-label classification," *Pattern Recognition*, vol. 118, p. 107965, 2021.

[15] N. Rastin, M. Taheri, and M. Z. Jahromi, "A stacking weighted k-Nearest neighbour with thresholding," *Information Sciences*, vol. 571, pp. 605–622, 2021.

[16] K. Napierala and J. Stefanowski, "Types of minority class examples and their influence on learning classifiers from imbalanced data," *J. Intell. Inf. Syst.*, vol. 46, no. 3, pp. 563–597, 2016.

[17] T. Jo and N. Japkowicz, "Class imbalances versus small disjuncts," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, p. 40–49, jun 2004.

[18] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113–141, 2013.

[19] J. F. Díez-Pastor, J. J. Rodríguez, C. I. García-Osorio, and L. I. Kuncheva, "Random balance: Ensembles of variable priors classifiers for imbalanced data," *Knowledge-Based Systems*, vol. 85, pp. 96–111, 2015.

[20] H. G. Zefrehi and H. Altınçay, "Imbalance learning using heterogeneous ensembles," *Expert Systems with Applications*, vol. 142, 2020.

[21] J. Gong and H. Kim, "RHSBOOST: Improving classification performance in imbalance data," *Computational Statistics and Data Analysis*, vol. 111, pp. 1–13, 2017.

[22] M. Koziarski, "Radial-based undersampling for imbalanced data classification," *Pattern Recognition*, vol. 102, p. 107262, 2020.

[23] C. X. Ling and C. Li, "Data mining for direct marketing: Problems and solutions," in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, ser. KDD'98. AAAI Press, 1998, p. 73–79.

[24] A. Fernandez, S. Garcia, F. Herrera, and N. V. Chawla, "SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary," *Jornal of Artificial Intelligence Research*, vol. 61, pp. 863–905, 2018.

[25] G. Kovács, "An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets," *Applied Soft Computing*, vol. 83, p. 105662, 2019.

[26] C. Chen, A. Liaw, and L. Breiman, "Using random forest to learn imbalanced data," Department of Statistics, UC Berkley, Technical Report 666, 2004.

[27] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data." *IEEE Trans. Systems, Man, and Cybernetics*, vol. 2, no. 3, pp. 408–421, 1972. [Online]. Available: http://dblp.uni-trier.de/db/journals/tsmc/tsmc2.html#Wilson72

[28] P. E. Hart, "The condensed nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 14, pp. 515–516, 1968.

[29] I. Tomek, "Two Modifications of CNN," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7(2), pp. 679–772, 1976.

[30] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme, "Cost-sensitive learning methods for imbalanced data," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 2010, pp. 1–8.

[31] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proceedings of the Fifth ACM SIGKDD Internaional Conference on Knowledge Discovery and Data Mining*, 11 2002.

[32] K. Veropoulos, C. Campbell, and N. Cristianini, "Controlling the sensitivity of support vector machines," in *Proceedings of the International Joint Conference on AI*, 1999, pp. 55–60.

[33] S. Lessmann, "Solving imbalanced classification problems with support vector machines," in *Proceedings of the International Conference on Artificial Intelligence, IC-AI '04, June 21-24, 2004, Las Vegas, Nevada, USA, Volume 1*, H. R. Arabnia, Ed. CSREA Press, 2004, pp. 214–220.

[34] C. Wang, C. Deng, and S. Wang, "Imbalance-XGBoost: leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost," *Pattern Recognition Letters*, vol. 136, pp. 190–197, 2020.

[35] K. M. Ting, "Inducing cost-sensitive trees via instance weighting," in *Principles of Data Mining and Knowledge Discovery*, J. M. Żytkow and M. Quafafou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 139–147.

[36] Xiaoyong Chai, Lin Deng, Qiang Yang, and C. X. Ling, "Test-cost sensitive naive Bayes classification," in *Fourth IEEE International Conference on Data Mining (ICDM'04)*, 2004, pp. 51–58.

[37] M. Kukar and I. Kononenko, "Cost-sensitive learning with neural networks," in *ECAI, 13th European Conference on Artificial Intelligence*, 1998.

[38] J. M. Johnson and T. M. Khoshgoftaar, "Deep learning and thresholding with class-imbalanced big data," in *Proceedings of 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019.

[39] Y. Yang, "A study on thresholding strategies for text categorization," in *Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval*. ACM Press, 2001, pp. 137–145.

[40] A. Sun, E.-P. Lim, and Y. Liu, "On strategies for imbalanced text classification using SVM: A comparative study," *Decision Support Systems*, vol. 48, no. 1, pp. 191 – 201, 2009.

[41] H. Yu, C. Mu, C. Sun, W. Yang, X. Yang, and X. Zuo, "Support vector machine-based optimized decision threshold adjustment strategy for classifying imbalanced data," *Knowledge-Based Systems*, vol. 76, pp. 67 – 78, 2015.

[42] Z. C. Lipton, C. Elkan, and B. Naryanaswamy, "Optimal thresholding of classifiers to maximize F1 measure," in *Machine Learning and Knowledge Discovery in Databases*, T. Calders, F. Esposito, E. Hüllermeier, and R. Meo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 225–239.

[43] G. Collell, D. Prelec, and K. R. Patil, "A simple plug-in bagging ensemble based on threshold-moving for classifying binary and multiclass imbalanced data," *Neurocomputing*, vol. 275, pp. 330–340, 2018.

[44] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[45] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano, "RUSBoost: A hybrid approach to alleviating class imbalance," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 40, no. 1, pp. 185–197, 2010.

[46] X. Y. Liu, J. Wu, and Z. H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 539–550, 2009.

[47] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, Nov 2016.

[48] M. Lango and J. Stefanowski, "What makes multi-class imbalanced problems difficult? An experimental study," *Expert Systems with Applications*, vol. 199, p. 116962, 2022.

[49] L. I. Kuncheva, A. Arnaiz-Gonzalez, J. Diez-Pastor, and I. A. D. Gunn, "Instance selection improves geometric mean accuracy: A study on imbalanced data classification," *Progress in AI*, vol. 8, no. 2, pp. 215–228, 2019.

[50] J.-F. Díez-Pastor, J. Rodríguez, C. García-Osorio, and L. Kuncheva, "Diversity techniques improve the performance of the best imbalance learning ensembles," *Information Sciences*, vol. 325, pp. 98–117, 12 2015.

[51] D. A. Cieslak and N. V. Chawla, "Learning decision trees for unbalanced data," pp. 241–256, 2008.

[52] W. Liu, S. Chawla, D. A. Cieslak, and N. V. Chawla, "A robust decision tree algorithm for imbalanced data sets," pp. 766–777, 2010.

[53] J. R. Quinlan, "Improved estimates for the accuracy of small disjuncts," *Machine Learning*, vol. 6, no. 1, pp. 93–98, Jan 1991.

[54] B. Zadrozny and C. Elkan, "Learning and making decisions when costs and probabilities are both unknown," pp. 204–213, 2001.

[55] Q. Zou, S. Xie, Z. Lin, M. Wu, and Y. Ju, "Finding the best classification threshold in imbalanced classification," *Big Data Research*, vol. 5, pp. 2–8, 2016.

[56] H. Guo and H. L. Viktor, "Learning from imbalanced data sets with boosting and data generation: The databoost-IM approach," *SIGKDD Explorations*, vol. 6, no. 1, pp. 30–39, Jun. 2004.

[57] C. X. Ling, V. S. Sheng, and Q. Yang, "Test strategies for cost-sensitive decision trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 8, pp. 1055–1067, 2006.

[58] K. Veropoulos, C. Campbell, and N. Cristianini, "Controlling the sensitivity of support vector machines," in *Proceedings of the International Joint Conference on AI*, 1999, pp. 55–60.

[59] J. J. Chen, C. A. Tsai, H. Moon, H. Ahn, J. J. Young, and C. H. Chen, "Decision threshold adjustment in class prediction," *SAR and QSAR in Environmental Research*, vol. 17, no. 3, pp. 337–352, 2006.

[60] W.-J. Lin and J. Chen, "Class-imbalanced classifiers for high-dimensional data," *Briefings in bioinformatics*, vol. 14, no. 1, pp. 13–26, 03 2012.

[61] Zhi-Hua Zhou and Xu-Ying Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2006.

[62] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.

[63] M. Saerens, P. Latinne, and C. Decaestecker, "Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure," *Neural Computation*, vol. 14, pp. 21–41, 02 2002.

[64] L. Tang, S. Rajan, and V. K. Narayanan, "Large scale multi-label classification via metalabeler," in *Proceedings of the 18th International Conference on World Wide Web*, ser. WWW '09. New York, USA: Association for Computing Machinery, 2009, p. 211–220.

[65] M. Ioannou, G. Sakkas, G. Tsoumakas, and I. Vlahavas, "Obtaining bipartitions from score vectors for multi-label classification," in *22nd IEEE International Conference on Tools with Artificial Intelligence*, vol. 1, 2010, pp. 409–416.

[66] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Proceedings of the 14th International Conference on Neural Information Processing Systems*, ser. NIPS'01, 2001, p. 681–687.

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

[68] S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, pp. 405–425, 2014.

[69] C. Halimu and A. Kasem, "A novel ensemble method for classification in imbalanced datasets using split balancing technique based on instance hardness (sBal_IH)," *Neural Computing and Applications*, vol. 33, no. 17, pp. 11 233–11 254, 2021.

[70] Y. Xie, M. Qiu, H. Zhang, L. Peng, and Z. Chen, "Gaussian distribution based oversampling for imbalanced data classification," *IEEE Transactions on Knowledge & Data Engineering*, vol. 34, no. 02, pp. 667–679, feb 2022.

[71] R. Blagus and L. Lusa, "SMOTE for high-dimensional class-imbalanced data," *BMC Bioinformatics*, vol. 14, no. 106, 2013.

[72] A. S. Tarawneh, A. B. A. Hassanat, K. Almohammadi, D. Chetverikov, and C. Bellinger, "SMOTEFUNA: Synthetic minority over-sampling technique based on furthest neighbour algorithm," *IEEE Access*, vol. 8, pp. 59 069–59 082, 2020.

[73] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "DBSMOTE: Density-based synthetic minority over-sampling technique," *Applied Intelligence*, vol. 36, no. 3, pp. 664–684, 2012.

[74] Y. Liang, S. Hu, L. Ma, and Y. He, "MSMOTE: Improving classification performance when training data is imbalanced," in *Computer Science and Engineering, International Workshop on*, vol. 2. Los Alamitos, CA, USA: IEEE Computer Society, oct 2009, pp. 13–17.

[75] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *Advances in Knowledge Discovery and Data Mining*, T. Theeramunkong, B. Kijsirikul, N. Cercone, and T.-B. Ho, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 475–482.

[76] J. Hu, X. He, D. J. Yu, and H. B. S. X. B. Yang, J. Y. Yang, "A new supervised over-sampling algorithm with application to protein-nucleotide binding residue prediction," *PLOS ONE*, vol. 9, no. 9, pp. 1–10, 2014.

[77] G. Douzas and F. Bacao, "Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE," *Information Sciences*, vol. 501, pp. 118–135, 2019.

[78] M. Koziarski, B. Krawczyk, and M. Woźniak, "Radial-based oversampling for noisy imbalanced data classification," *Neurocomputing*, vol. 343, pp. 19–33, 2019.

[79] X. Tao, Q. Li, W. Guo, C. Ren, Q. He, R. Liu, and J. Zou, "Adaptive weighted over-sampling for imbalanced datasets based on density peaks clustering with heuristic filtering," *Information Sciences*, vol. 519, pp. 43–73, 2020.

[80] G. Menardi and N. Torelli, "Training and assessing classification rules with imbalanced data," *Data Mining and Knowledge Discovery*, vol. 28, pp. 92–122, 2014.

[81] C. Bellinger, S. Sharma, N. Japkowicz, and O. R. Zaïane, "Framework for extreme imbalance classification: SWIM—sampling with the majority class," *Knowledge and Information Systems*, vol. 62, pp. 841–866, 2020.

[82] H. Haibo, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *IEEE International Joint Conference on Neural Networks (IEEE WorId Congress on Computational Intelligence)*, 2008, pp. 1322–1328.

[83] K. Li, W. Zhang, Q. Lu, and X. Fang, "An improved smote imbalanced data classification method based on support degree," in *International Conference on Identification, Information and Knowledge in the Internet of Things*, 2014, pp. 34–38.

[84] J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera, "SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering," *Information Sciences*, vol. 291, pp. 184–203, 2015.

[85] K. Puntumapon and K. Waiyamai, "A pruning-based approach for searching precise and generalized region for synthetic minority over-sampling," in *Advances in Knowledge Discovery and Data Mining, Springer Berlin Heidelberg*, 05 2012, pp. 371–382.

[86] S. Gazzah, N. Essoukri, and B. Amara, "New over-sampling approaches based on polynomial fitting for imbalanced datasets," pp. 411–458, 2008.

[87] L. Jiang, C. Qiu, and C. Li, "A novel minority cloning technique for cost-sensitive learning," *Internations Journal of Pattern Recognition and Artificial Intelligence*, vol. 29, pp. 1 551 004:1–1 551 004:18, 2015.

[88] F. R. Torres, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "Smote-d a deterministic version of smote," in *Pattern Recognition*, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, V. Ayala Ramirez, J. A. Olvera-López, and X. Jiang, Eds. Cham: Springer International Publishing, 2016, pp. 177–188.

[89] D. Cieslak, N. Chawla, and A. Striegel, "Combating imbalance in network intrusion datasets," in *2006 IEEE International Conference on Granular Computing*, 2006, pp. 732–737.

[90] M. Li and S. Fan, "CURE-SMOTE algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests," *BMC Bioinform.*, vol. 18, no. 1, pp. 169:1–169:18, 2017.

[91] L. Abdi and S. Hashemi, "To combat multi-class imbalanced problems by means of over-sampling techniques." *IEEE Transactions Knowledge and Data Engineering*, vol. 28, no. 1, pp. 238–251, 2016.

[92] W. Siriseriwan and K. Sinapiromsaran, "Adaptive neighbor synthetic minority oversampling technique under 1NN outcast handling," *Songklanakarin Journal of Science and Technology*, vol. 39, no. 5, pp. 565–576, 2017.

[93] G. Douzas, F. Bacao, and F. Last, "Improving imbalanced learning through a heuristic oversampling method based on K-Means and SMOTE," *Information Sciences*, vol. 465, no. C, p. 1–20, oct 2018.

[94] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Advances in Intelligent Computing*, D.-S. Huang, X.-P. Zhang, and G.-B. Huang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 878–887.

[95] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, p. 20–29, jun 2004.

[96] M. Koziarski and M. Wozniak, "Ccr: A combined cleaning and resampling algorithm for imbalanced data classification," *International Journal of Applied Mathematics and Computer Science*, vol. 27, pp. 727 – 736, 2017.

[97] B. Krawczyk, M. Woźniak, and F. Herrera, "Weighted one-class classification for different types of minority class examples in imbalanced data," in *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 2014, pp. 337–344.

[98] J. Błaszczyński and J. Stefanowski, "Neighbourhood sampling in bagging for imbalanced data," *Neurocomputing*, vol. 150, pp. 529–542, 2015.

[99] J. F. Díez-Pastor, J. J. Rodríguez, C. I. García-Osorio, and L. I. Kuncheva, "Diversity techniques improve the performance of the best imbalance learning ensembles," *Information Sciences*, vol. 325, pp. 98–117, 2015.

[100] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, and Y. Zhou, "A novel ensemble method for classifying imbalanced data," *Pattern Recognition*, vol. 48, no. 5, pp. 1623–1637, 2015.

[101] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrer, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems Man and Cybernetics Part C*, vol. 42, no. 4, pp. 463–484, Jul. 2012.

[102] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing,

"Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, 2017.

[103] A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating probability with undersampling for unbalanced classification," in *IEEE Symposium Series on Computational Intelligence, SSCI2015, Cape Town, South Africa*, 2015, pp. 159–166.

[104] B. Wallace and I. Dahabreh, "Improving class probability estimates for imbalanced data," *Knowledge and Information Systems*, vol. 41, pp. 33–52, 10 2014.

[105] G. Katz, E. C. R. Shin, and D. Song, "ExploreKit: Automatic feature generation and selection," in *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, Barcelona, Spain*, F. Bonchi, J. Domingo-Ferrer, R. Baeza-Yates, Z. Zhou, and X. Wu, Eds. IEEE Computer Society, 2016, pp. 979–984.

[106] Z. Erenel and H. Altınçay, "Improving the precision-recall trade-off in undersampling-based binary text categorization using unanimity rule," *Neural Computing and Applications*, vol. 22, no. S1, pp. 83–100, Jan. 2013.

[107] R. Soleymani, E. Granger, and G. Fumera, "F-measure curves: A tool to visualize classifier performance under imbalance," *Pattern Recognition*, vol. 100, p. 107146, 2020.

[108] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, pp. 255–287, 2011.

[109] W. Siriseriwan and K. Sinapiromsaran, "Adaptive neighbor synthetic minority oversampling technique under 1nn outcast handling," *Songklanakarin Journal of Science and Technology*, vol. 39, pp. 565–576, 09 2017.

[110] K. M. Ting, J. R. Wells, S. C. Tan, S. W. Teng, and G. I. Webb, "Feature-subspace aggregating: ensembles for stable and unstable learners," *Machine Learning*, vol. 82, no. 3, pp. 375–397, Mar. 2011.

[111] M. Skurichina and R. P. W. Duin, "Bagging, Boosting and the Random Subspace Method for Linear Classifiers," *Pattern Analysis & Applications*, vol. 5, no. 2, pp. 121–135, Jun. 2002.

[112] T. G. Dietterich, "An Experimental Comparison of Three Methods for

Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization," *Machine Learning*, vol. 40, pp. 139–157, 2000.

[113] L. Breiman, "Bias, variance and arcing classifiers," *Technical Report 460, Statistics Department, Berkeley*, 1996. [Online]. Available: https://www.bibsonomy.org/bibtex/265f179a69a81cebd376b94f71f35b31d/brefeld

[114] E. Bauer and R. Kohavi, "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants," *Machine Learning*, vol. 36, pp. 105–139, 1999.

[115] M. Skurichina and R. P. W. Duin, "Boosting in linear discriminant analysis," in *Proceedings of the First International Workshop on Multiple Classifier Systems*. Berlin, Heidelberg: Springer-Verlag, 2000, p. 190–199.

[116] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, Dec. 2006.