GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

ISSN 1612-6793

Bachelor's Thesis

submitted in partial fulfillment of the
requirements for the course "Applied Computer Science"

Data Classification in Medical/Healthcare: Classification Methods Comparison under Class Imbalance

Mengru Ji

Institute of Computer Science

Bachelor's and Master's Theses
of the Center for Computational Sciences
at the Georg-August-Universität Göttingen

13. March 2021

Georg-August-Universität Göttingen
Institute of Computer Science

Goldschmidtstraße 7
37077 Göttingen
Germany

☎ +49 (551) 39-172000
FAX +49 (551) 39-14403
✉ office@informatik.uni-goettingen.de
🌍 www.informatik.uni-goettingen.de

First Supervisor:	Prof. Dr. Xiaoming Fu
Second Supervisor:	Dr. Tingting Yuan

I hereby declare that I have written this thesis independently without any help from others and without the use of documents or aids other than those stated. I have mentioned all used sources and cited them correctly according to established academic citation rules.

Göttingen, 13. March 2021

Abstract

Recently, machine learning has become a popular topic in the computer field and classification is one of the most prevalent directions in machine learning. There are many standard and commonly used classification methods, such as Naïve Bayes, k-nearest neighbor (kNN), support vector machine and decision tree. Most classification algorithms assume that the dataset is balanced distributed, but data obtained from real life can hardly meet this prerequisite. In the real world, the impact of data imbalance in many domains is pervasive, such as in fraud detection, risk management, text classification and medical diagnosis. Especially in the medical or healthcare sector, the cost of missing a patient is much more expensive than that of classifying a healthy person as a patient. If this problem is ignored, it will not be easy to guarantee the accuracy of the classification task.

Due to the prevalence of class imbalance problem in medical or healthcare sector, in order to have a comprehensive and deep understanding of the imbalance classification algorithms, in this paper, a comparison of the binary classification performance of several classical and state-of-the-art class imbalance classification algorithms was carried out on several datasets. The data used for the experiments was taken from UCI, KEEL and OpenML which included not only the data from the medical/healthcare sector but also some other widely used datasets.

The algorithms used for the comparison included DDAE, MWMOTE, SMOTE, RUSBoost, AdaBoost, cost-sensitive decision tree(csDCT), self-paced Ensemble Classifier, MetaCost, CAdaMEC and Iterative Metric Learning (IML), most of which come from three main groups of class imbalance classification, namely sampling, cost-sensitive learning and ensemble learning. The experiments mainly focused on an overall comparison, the impact of imbalance ratio and the size of the given dataset on the performance of the above-mentioned algorithms. Mainly DDAE is researched, including the effectiveness of each component and the impact of parameters upon it. Sampling methods are not suitable all the time since they need to consider the neighborhoods based on distance. However, some classifiers can be improved after the balance of class distribution. Cost-sensitive learning models should be utilized when the dataset is less in imbalance, because it is difficult to set an appropriate cost matrix for the specific dataset, which can cause the fluctuations of their performances. Ensemble learning techniques perform better as they approach the problem from many angles, but they absorb not only the advantages but also the disadvantages of the techniques used. In addition, according to the results, the Data Block Construction (DBC) was the most important component in DDAE. Also, its parameter tuning process showed as the most appropriate parameter for the model, especially in the experiment on the number of data blocks of DDAE, which illustrates the impact of the number of base learners on the performance of ensemble learning algorithms.

Acknowledgements

Here, I would like to sincerely acknowledge individuals who have supported me directly or indirectly for the accomplishment of this paper.

First of all, I would like to express my sincere gratitude to my first supervisor Prof. Dr. Xiaoming Fu for his helpful suggestions for me and for my paper. Secondly, I would like to thank Dr. Tingting Yuan for being my second supervisor and guiding me with her comments. Next, I would like to specially thank Mr. Yachao Shao, who has been my advisor throughout the journey towards completion of my paper. I would like to thank him for his constant encouragement, guidance and tireless assistance. Without their usual and enlightening guidance, this paper would not have been in its present form.

I would like to thank my dear family for their love and great faith in me for many years, even though they cannot be beside me. I also want to thank my dear friends and my roommate, who have given me help and time to listen to my opinions and comforted me when I felt upset or faced difficulties. While it would be difficult to list all their names, I would like to express my gratitude to each of my family members and friends in China and in Germany. Without their support and help, the completion of this paper would not have been possible.

Finally, I would like to thank the faculty of Computer Science of the Georg-August University of Göttingen. I would like to thank all the staff members of the department who have provided me with information required and answered my questions with patience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In recent times, machine learning has advanced briskly and has become an attractive research topic in the domain of computer science. The classification problem is one of the most prevalent directions of machine learning [3], and many researchers have paid attention to binary imbalance classification [4]. There are many standard and commonly used classification methods that are already known, such as C4.5 [5], support vector machine [6], k-nearest neighbor(kNN) [7] and random forest [8]. Most algorithms used for classification consider the datasets are to be balanced distributed, but data obtained from real life can hardly meet this prerequisite [9]. The class imbalance of the data will impact on the performance of almost all standard classification algorithms [3]. Since the number of instances of one class is more than that of another class, the model is likely to be confused by the majority during the learning process so that it does not perform well in the final classification of the test data [10]. In the real world, the impact of data imbalance in many fields is pervasive, such as medical diagnosis, fraud detection and text classification [9]. This is especially true in the medical or healthcare field, because patients with a particular disease themselves belong to a minority in society and because of the complexity of medical information, the process of collecting information is prone to record errors or data omissions [11, 12]. If this problem is ignored, it will not be easy to guarantee accuracy when using classification algorithms to predict testing data [10]. This is not only a waste of resources, but it can also lead to misdiagnosis due to incorrect predictions, which will endanger the lives of patients in severe cases [13, 14]. Therefore, this problem has critical importance and its probability of occurrence is very high.

According to previous research, the imbalance problem of classes can be improved mainly in the following areas: (1) Sampling technique [15]: including oversampling and undersampling; (2) Cost-sensitive learning [16]: where the value of the majority class is often different from that of the minority class which also shows that the cost of predicted minority samples as the majority sample is more expensive than that of predicted majority sample as the minority sample [13]; and, (3) Ensemble learning [17] inspired by the popular saying that the minority obeys the majority, that is, Major Voting. A certain number of basic classifiers will be used to predict all anonymous data samples, and then the final label of a particular sample will be determined according to the number of votes [18].

In the past two decades, assorted models have been developed to address the problem of class imbalanced classification. Most of them are the application of the three ideas mentioned in the previous paragraph or a combination of them. For example, the SMOTE [19] belonging to the field of resampling, the Adacost [20] algorithm utilizing both cost-sensitive learning and the ensemble learning technique, and the application of ensemble learning in RUSBoost [21]. These are classical algorithms in the field of imbalanced learning, and have been applied to process data from multiple fields. In addition, there are many algorithms that focus on data imbalance in the medical and health fields, such as [11, 22, 23].

Due to the variety of imbalance classification algorithms, it can be hard to see the differences between them. To understand them better, this paper focuses on comparing the performance of different types of imbalance classification algorithms on multiple datasets which are not only from medical/healthcare sector but also from other popular fields. There are seven classical imbalanced classification algorithms, including (1) sampling: SMOTE [19] and MWMOTE [24]; (2) cost-sensitive learning: MetaCost [25], CAdaMEC [26] and cost-sensitive decision tree [27]; and (3) ensemble learning: AdaBoost [28] and RUSBoost [21], with three newer state-of-art models, namely DDAE [29], Iterative Metric Learning(IML) [30] and self-paced Ensemble Classifier [31], developed in recent years and selected as a comparison. The experiment first analyzes the performance of different models based on their evaluation metrics on the same dataset. Secondly, the impact of other factors, such as the size of the dataset and the imbalance ratio, are illustrated. In the final part of the experiment, the importance of the various components of DDAE is presented.

1.2 Outline of Thesis

This paper is structured as follows. The background and objective are presented in Chapter 1. Chapter 2 focuses on the previous works on the class imbalance problem and some of the classification methods applied in the medical/healthcare sector. In Chapter 3, the methodology containing detail about the implementation DDAE model and IML model is described. Chapter 4 shows input data and presents the result of experiment. The overall results are discussed and concluded in Chapter 5.

Chapter 2

Literature Review

In section 2.1, some theories related to imbalanced datasets are described, including their importance, the impact of imbalanced medical datasets, and a review of related works that focused on imbalanced medical datasets. This section highlights the gaps in the works on imbalanced data classification if only general classification algorithms are used. It is followed by the section 2.2 which includes the methods of dealing with imbalanced classification in detail from three perspectives and several state-of-the-art and classical algorithms for imbalanced classification are introduced and explained. At the end of this chapter, in section 2.3, several metrics applied for evaluating the imbalanced classification models are briefly described.

2.1 An Overview of Theory Related to Imbalanced Medical Datasets

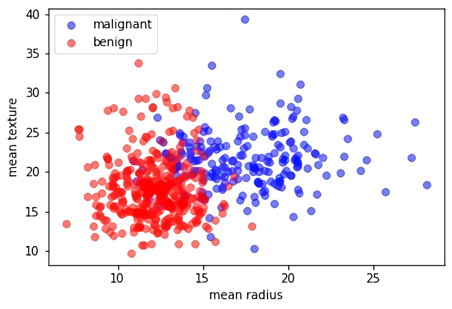


Figure 2.1: Class distribution on the dataset with IR=1.684

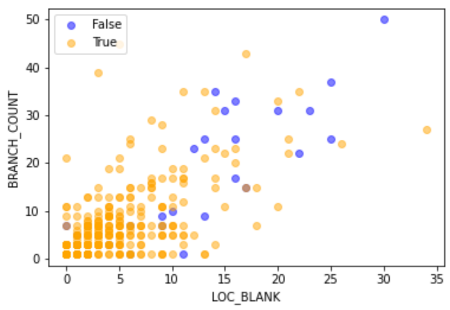


Figure 2.2: Class distribution on the dataset with IR=12

The class imbalance problem means one class contains a small number of data instances, but the other one is represented by a large number of data instances [27]. In practical situations, the ratio between these two classes may be substantial, similar to 1:100, 1:1000, 1:10000, or even higher [32]. Figure 2.1 and Figure 2.2 show the difference between two datasets with an Imbalance Ratio(IR) of 12 and 1.684, respectively. In addition, it should be noted that, under some specific circumstances, the problem of class imbalance has an internal cause [32]. For example, Haibo He and Edwardo A. Garcia gave an example of cancer classification in their 2009 work [10]. Such an example demonstrates that, in the screened population, the prevalence of cancer is particularly low, which leads to the class imbalance problem in the collected data, in which one disease state is underrepresented [10]. However, due to the limitation of the data collection process, in areas where there is no inherent imbalance, there is also the possibility for the occurrence of a class imbalance problem: namely, the way of choosing the data by researchers (for example which data and in what quantity) and the imbalanced cost for different errors can also lead to class imbalance problems [32]. Such a situation may vary from case to case [32]. Affected by these conditions, normal classifiers are often confused by the majority class and ignore the minority class [32]. Additionally, for the percentage of examples available for each class, most real-world data processed using non-linear classification strategies are imbalanced, which may cause the algorithm to learn overly complicated models that overfit the data and have almost no correlation [33]. This issue is critical since it leads to a significant barrier in the performance achieved by basic learning methods which assume that the class distribution is balanced [33].

Classification in medical diagnostics can aid in disease diagnosis and predict outcomes in response to treatment [22]. For instance, with Computer Aided Decision(CAD) systems, a physician is able to diagnose a patient with the help of computer algorithms and classification is one of the most common tasks performed by the CAD system [34]. However, data collection in the medical sector is associated with a number of challenges and practical limitations. The collection of patient data is time-consuming [34] and the collection of balanced model training data is difficult where there is low prevalence of the disease [12]. In addition, the inherent heterogeneity, incompleteness, and high-dimensional nature of healthcare/medical data can also lead to challenges [11]. The medical data is often heterogeneous where patient records contain various data types, including images, real and integer with different ranges, and text types [11].

Many classification algorithms have been developed recently and used to detect and predict some common diseases such as diabetes, Parkinson's disease and vertebral column pathologies, which has brought significant trouble and pain to a tremendous number of patients [35].

Study [11] addressed the problems of imbalanced healthcare data on the diagnosis of a brain tumor. In this study, a new approach to data mining is adopted, combining feature selection and ensemble classification. Study [22] introduced a novel voting class weight algorithm called CWsRF based on random forest algorithm (RF), which focuses on the challenge of identifying the minority class sufficiently in medical applications and can be applied to the detection of diseases, such as breast cancer, or medical images classification. Study [23] offers an innovative ensemble learning paradigm for the early detection of diseases with imbalanced data that functions equally as well as or outperforms the other state-of-the-art comparison algorithms, such as support vector machines and random forest. This algorithm is the first comprehensive ensemble learning technique that involves multiple SVM diversity structures for classification, which can prevent the generation of noisy instances and effectively re-balance the input data [23].

2.2 Related Work on Imbalanced Classification

2.2.1 Sampling Methods

Resampling, which involves creating a new transformed version of the training set of an imbalanced dataset, offers a set of practical and straightforward approaches to provide a more balanced data distribution [15]. All these approaches can be classified into three groups: Oversampling techniques, Undersampling techniques, and Hybrids techniques [27]. In the first case, oversampling, the original dataset will be augmented by replicating the selected instances or creating new instances from the existing one, while with undersampling methods, a set of data (usually majority class samples) will be removed from the original set. The hybrid methods are a combination of both from sampling algorithms [27]. Among these three groups, random oversampling and random undersampling are non-heuristic and the simplest approaches; the former case has the drawback of overfitting, and in the case of the latter, the omission of some crucial samples pertaining to the majority class may occur due to the removal of examples [10]. Moreover, informed undersampling based on the traditional undersampling method aims to deal with information loss deficiency. The *EasyEnsemble* and *BalanceCasade* are two applications of this algorithm with good performance [36]. Additionally, synthetic sampling with data generation is also a useful approach to improve data distribution. Some of the renowned algorithms in this area are the *SMOTE* [19] and the *Borderline-SMOTE* [37], which was applied to various situations with a great deal of success. In addition, to deal with the overlapping introduced from sample methods, data cleaning techniques have been used practically [10]. *Tomek links* [38] is representative of this kind of method.

2.2.2 Cost-Sensitive Learning Methods

Sampling methods try to improve the balanced level of the dataset, while cost-sensitive learning methods are utilized to deal with different misclassification errors that incur different penalties to find the optimal decision. The foundation of these decisions is the cost matrix, which is fundamental to the cost-sensitive learning methodology [10, 13]. Table 2.1 illustrates the structure of a binary classification cost matrix. The positive category (class label 1) denotes the minority and the negative category (class label 0) represents the majority. If m stands for the predicted label and n stands for the actual label, so the $C(m, n)$ is the cost of predicting a class n sample as class m. For example, $C(1, 0)$ represents the cost of predicting a negative instance as positive. In view of the cost matrix, the purpose of this type of learning method is explained to create a model with minimal overall misclassification costs [16,39].

	Actual negative	Actual positive
Predict negative	$C(0,0)$	$C(0,1)$
Predict positive	$C(1,0)$	$C(1,1)$

Table 2.1: Cost Matrix for Binary Classification

Bearing in mind most traditional classifiers assume that the misclassification has the same cost for false negative(FN) and false positive(FP) [13]. However, real-world scenarios are not so ideal. Conceptually, in certain situations, the cost of incorrect labeling for a sample should always be higher than a correct one [13]. For instance, in cancer diagnosis, where a patient who does have a certain kind of cancer is classified as negative(FN), this error could be much more serious and expensive than regarding a non-cancer patient as positive(FP), which can be corrected by further medical examinations; the former situation may lead to a worse patient condition or the even more severe outcome of losing his life [14]. In another scenario, the cost of not mailing to potential buyers is higher than mailing to some non-buyers [16]. From this, it can also be illustrated that costs are not necessarily only monetary but also can be the severity of an illness or the wasting of time [13].

The cost matrix values should be defined with care because the supplied cost matrix can strongly impact effectiveness [27]. However, in many cases, the cost of classification errors cannot be described clearly. The determination of a given domain's cost representation can be a challenge and, under some circumstances, impossible [10]. There may be a problematic accession to a domain expert or a lack of available prior information on the cost matrix during the classifiers' training process, which is a common phenomenon when cost-sensitive learning is utilized to deal with the class imbalance problem [27].

Cost-sensitive learning techniques can be categorized into two families: *Meta-learning* (including two aspects: thresholding and sampling) and *Direct methods* [39]. The principle of the former category is to create a "wrapper" to turn existing cost-insensitive classifiers into cost-sensitive classifiers, and in the latter case, classifiers that are cost-sensitive in themselves are constructed [39].

The *cost-sensitive decision trees* which have taken account of misclassification during pruning are popularly used as direct methods for detecting card fraud when the cost to misclassify could vary [39,40]. *MetaCost* [25] can be viewed as a representative thresholding tool in the case of thresholding. *Weighting* [41] is one implementation of the sampling methods, in which examples of the minority are assigned high weights according to their proportion.

2.2.3 Ensemble Learning Methods

In 2004, a statistician named James Michael Surowiecki presented in his study [42] that, under certain controlled circumstances, the decisions or predictions made by groups of humans often outperform those made by a single individual.

The ensemble methodology is used in classification to enhance the output of an individual classifier, which is likely to ask multiple "experts" for help: the key idea is to train multiple classifiers and then combine them to achieve an overall classification that overtakes each indivudual classification to produce the final decision [18]. Therefore, the predictions of all members of this ensemble, known as classifier fusion or aggregation are considered in the combining step, in order to classify a new unknown example [27]. In 1979, Dasarathy and Sheela proposed one of the earliest work on ensemble learning [43]. In [43], the partition of the feature space with two or more classifiers was discussed. Later in 1990, Hansen and Salmon introduced the idea that *ensemble artificial neural networks (ANNs)* with similar configurations can improve a single classifier's generalization performance [44]. Furthermore, the ensemble learning approach was successfully applied to various industries, including medical diagnosis [45], cheminformatics [46], bioinformatics [47] and so forth. *Bagging and Boosting* are the two most practical techniques of ensemble learning, which apply instance partitioning methods [48].

Bagging is the abbreviation of Bootstrap Aggregating introduced by Breiman in 1996 [49], which is a simple but successful method for building a set of classifiers that are suitable not only for dealing with binary, but also multi-class classification [17, 18]. In detail, every single classifier in the ensemble is trained on a set of examples which are taken with replacement from the training set [17, 18, 49]. The base classifier can be trained by using the base learning approach with each set [17]. Major voting is used to generate the ultimate prediction for the composite bagging classifier [18]. Study [17] indicates that, due to the generation of data samples with the bootstrap sampling [49], there is a large overlap among all data samples. A stable learner (a base algorithm that is intensive to perturbation on training sets) may lead to a set of classifiers whose predictions are very similar with no improvement in generalization after the combination [17, 18]. Therefore, a relatively unstable learner should be utilized to ensure the diversity among the ensemble classifiers under this scenario so that sufficiently different decision boundaries can be obtained for small perturbations in different training sets. Random Forests [8] is an extension of the bagging algorithm generated from individual decision trees, whose specific training parameters that can be bootstrapped replicas of the training data, as in bagging, vary randomly [50].

The word *boosting* refers to a group of algorithms which can transform weak learners (an algorithm generating classifiers that outperform random guessing) to strong learners(low error with high confidence for all class concepts), proven by Schapire in his 1990 study [51]. Similar to Bagging, boosting also generates an ensemble classifier by applying resampling methods on data and is later combined with major voting [18, 50]. Freund and Schapire introduced one of the representative works of boosting in 1997, named *AdaBoost*(Adaptive Boosting) [52], which applies an iterative process to simple boosting in order to improve performance. This approach focuses on the instances which are much more complex to classify. To be clear, in *AdaBoost*, a new dataset, in which more weight is assigned to the instances that are misclassified by the previous classifier and less weight is assigned to the one with a correct prediction, is used for training each subsequent classifier [53].

2.2.4 An Overview of Algorithms on Imbalanced Classification

Classical Imbalance Classification Algorithms

Synthetic Minority Oversampling Technique(SMOTE) [19] addresses the class imbalance problem by generating synthetic samples in feature space (Fig 2.3 SMOTE working procedure illustrates the detail of this procedure). One minority class example s_1 will be selected randomly and then its k nearest neighbors in minority class will be screened out; a line segment is formed between one of these k neighbors s_2, which is selected at random, and s_1 in the feature space [19]. SMOTE creates the synthetic samples through a convex combination of s_1 and s_2 [54]. As described in [55], random undersampling is suggested to be used to curtail the size of the majority in the first instance. Next, *SMOTE* is utilized on the training set to align the class distribution. This approach combined with random undersampling is proven to outperform the plain undersampling [19].

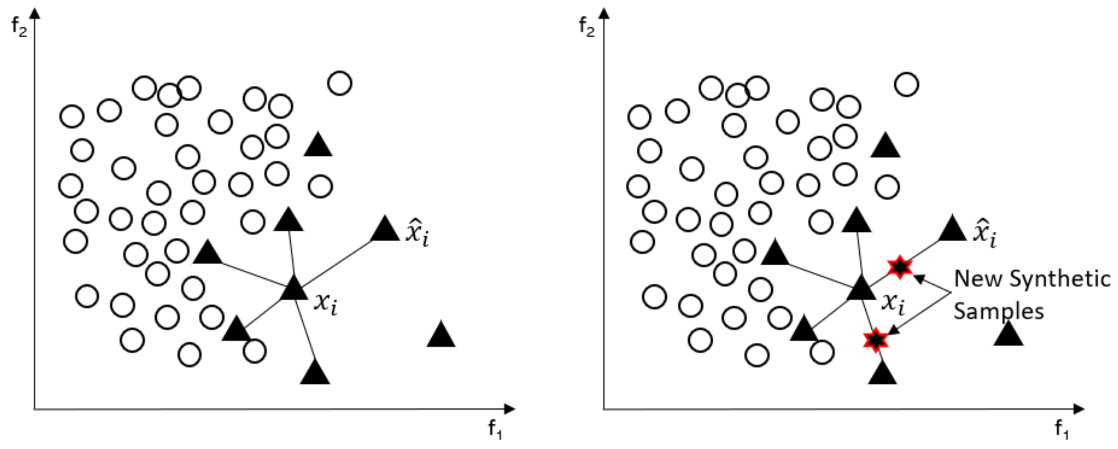


Figure 2.3: SMOTE working procedure [1]

AdaBoost [28] is a boosting ensemble learning approach utilized to deal with the class imbalance problem; the key principle behind it is to enhance the weak learner gradually into a strong learner. This is implemented by varying the sample weight, which indicates its importance in the classifier training process, stage by stage. Fig 2.4 shows the process of combining the ultimate classifier.

Majority Weighted Minority Oversampling Technique (MWMOTE) [24] balances the class distribution also by generating synthetic examples. Unlike SMOTE, this approach creates synthetic examples from a weighted minority class through a clustering approach; and the weight of each important minority sample is chosen on the basis of its Euclidean distance to the nearest majority class sample [24].

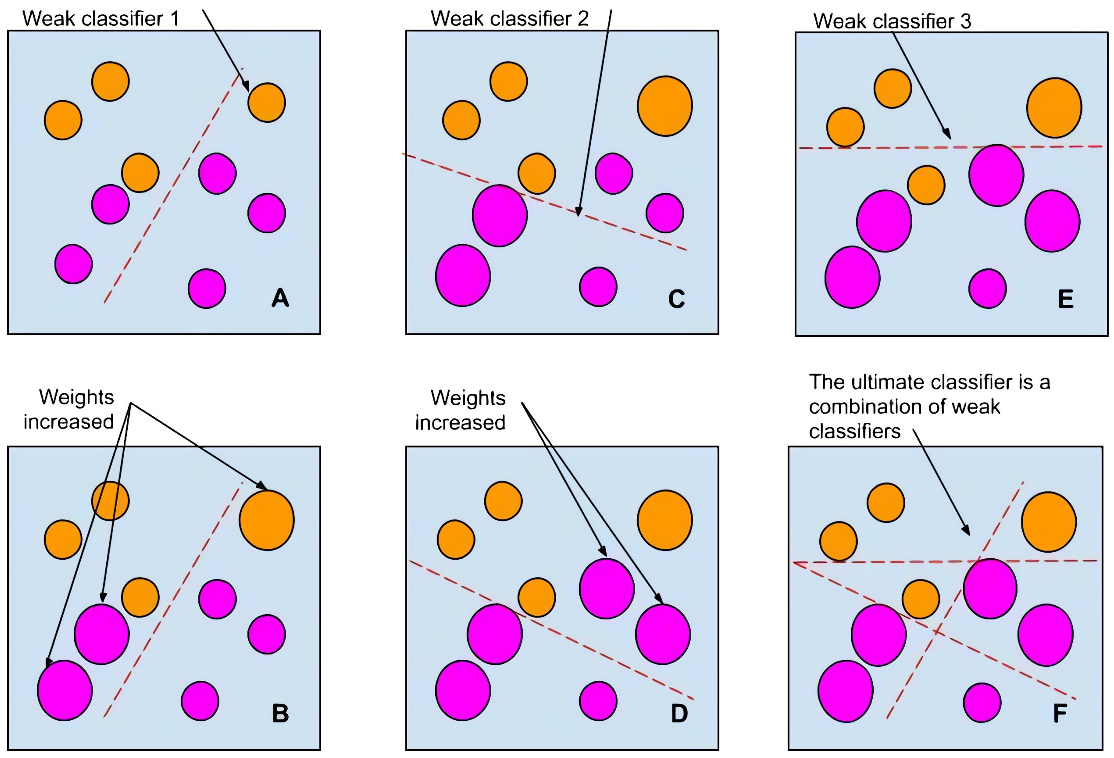


Figure 2.4: The process of the combination of the ultimate strong learner in *AdaBoost*

RUSBoost [21], a combination of data sampling and boosting algorithm, is based on *SMOTE-Boost* [56] that balances class distribution through *SMOTE* and works on improving classifier performance with the balanced data under the help of *AdaBoost*. Instead of *SMOTE*, this hybrid approach utilizes random sampling(RUS) to achieve increased performance [21].

CAdaMEC [26] is proposed upon *AdaMEC* [57](a cost-sensitive algorithm) through an appropriate calibration with Platt scaling.

MetaCost [25] is one of cost-sensitive learning models through wrapping a cost-minimizing method, which has no restriction on the number of classes or arbitrary cost matrices. This procedure relabels the instances in the training set with the class labels that have estimated minimal-cost, then the new replacement training set will be applied to the error-based learners [25].

The principle of ***cost-sensitive decision tree(csDCT)*** is to consider minimizing two separate costs: the test cost of the ith feature and the cost of misclassification of the sample [27].

Recent State-of-the-art Imbalance Classification Algorithms

Self-paced Ensemble Classifier [31] is an effective ensemble classifier generated by self-paced harmonizing data hardness through undersampling. It has been proven that this model can achieve robust performance even when the classes are highly overlapped and the data distribution highly skewed [31].

DDAE [29] is a novel model to address the class imbalance problem consisting of several classical approaches, including resampling, data metrics learning, cost-sensitive-learning, and ensemble learning. More detail will be described in section 3.2.1.

Iterative Metric Learning (IML) [30] focuses on the exploration of a stable neighborhood space for each of the data samples in the testing set. To achieve this, the proposed procedure utilizes an iterative metric learning technique [30]. More detail will be described in section 3.2.2.

2.3 Evaluation Metrics for Imbalanced Classification

Evaluation of classification performance always plays a significant role in guiding and learning performance [58]. ***Confusion matrix*** [59] provides a deeper understanding of predictive model performance through the results of the predictions so that the types of error can be more clearly observed. Table 2.2 shows the structure of a confusion matrix for binary classification. Table 2.3 shows the measures derived using the confusion matrix from Table 2.2

	Actual negative	Actual positive
Predict negative	TN(True Negative)	FN(False Positive)
Predict positive	FP(False Positive)	TP(True Negative)

Table 2.2: Confusion Matrix for Binary Classification

Cesar Ferri et al. categorized evaluation metrics into three groups in their 2008 work [60]: *probability metrics*, *ranking metrics* and *threshold metrics*. In this paper, two groups of metrics, threshold and ranking metrics, are applied to evaluate the model performance.

Thresholding metrics, which quantify the classification prediction error, focuses on the generalization ability of the trained classifier through the quality of the trained classifier when used to predict unknown examples [61]. The most common threshold metric is the ***accuracy*** of classification applied in most conventional applications; nevertheless, accuracy is inappropriate for evaluating the imbalanced dataset since it is simple for a classifier that only predicts the majority to yield a low error [9].

Sensitivity-specificity metrics are practical thresholding metrics for imbalanced classification that are applied by several researchers [62, 63]. As defined in Table 2.3, ***specificity*** means the true negative rate. ***Sensitivity***, the complement to specificity, describes the true positive rate.

The ***geometric mean(G-mean)*** calculated through the combination of sensitivity and specificity, which has been used by Kubat and Matwin in their 1997 paper [64], can balance both concerns. Sensitivity, Specificity and G-mean are taken into account when both positive and negative classes are meaningful at the same time [65]. In addition, Precision-Recall metrics arising from the fields of information retrieval are utilized when the output of the minority(class positive) is more crucial [66]. In this paper, ***F1***(the value of β in *F_β-Measure* is 1) is utilized as one of the most important evaluation metrics.

Metrics	Formula
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Error Rate	$\frac{FP+FN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall (Sensitivity)	$\frac{TP}{TP+FN}$
F_β-Measure	$\frac{(1+\beta^2) * \text{ Precision } * \text{ Recall}}{\beta^2 * \text{ Precision } + \text{ Recall}}$
Specificity	$\frac{TN}{TN+FP}$
Geometric Mean	$\sqrt{\text{Sensitivity } * \text{ Specificity}}$

Table 2.3: Evaluation Metrics based on Confusion Matrix

Nonetheless, threshold metrics are not suitable when the distribution of categories observed in the training dataset does not match the distribution of the test set and the actual data, which can make the performance misleading [54].

The ranking metrics focuses on how effectively the base classifiers rank the examples [60]. A numeric score of an example that refers to the probability of being classified as positive is provided by the base classifier, which shows the level of granularity instead of a simple prediction. Different thresholds whose choice affects the trade-offs of both classes' errors can be utilized to test classifiers' performance [27]. ***Receiver Operating Characteristics (ROC) Curve*** [67] is the most commonly applied ranking metric that is not based on a specific threshold. ROC Curve takes the true positive rate (TPR) and false positive rate (FPR) into account and each point of ROC Curve corresponds to the single classifier performance with a given distribution [10]. The ***area under the ROC curve (AUCROC)*** is generally applied to measure different classifiers' performance, which is summarized

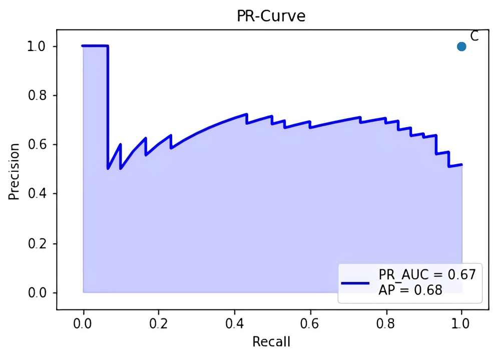


Figure 2.5: An example of PR Curve

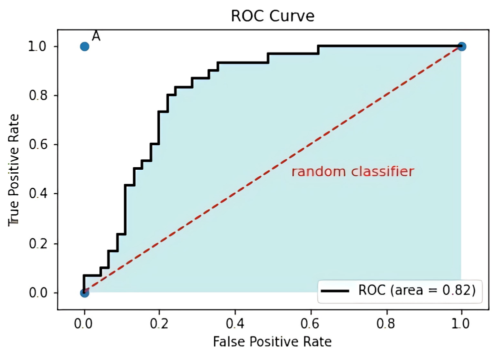


Figure 2.6: An example of ROC Curve

into a single metric [65]. An example of ROC Curve can be observed from Fig 2.6. Point A $(0, 1)$ represents the best performance of the classifier. Therefore, the closer the ROC curve is to A and the more it deviates from the 45-degree diagonal(representing a random classifier), the more successful it is; this also indicates the greater the AUCROC is, the better [65,67]. However, in [10], it is argued that even a classifier with a high AUCROC can perform poorly in a particular region in ROC space compared with a low AUCROC classifier.

If a dataset is highly skewed, the performance of the algorithm might as observed overly optimistic through a ROC curve [68]. The ***Precision-Recall (PR) Curve***, which assesses a more informative representation of performance, is utilized to solve such a limitation [10,69]. PR-Curve is a plot of Recall on the x-axis and Precision on the y-axis [69], and it can capture the performance of the classifier correctly and effectively if the number of false positives drastically change as the Precision metrics takes the ratio of TP to TP+FP into account [10]. Due to its high level of performances with highly skewed data, it has been applied to the evaluation of performance by many researchers, such as [70–72]. Unlike ROC-Curve, whose objective is to be closer to the point $(0, 1)$, the highest performing classifier is represented by a PR-Curve residing in the top right of the PR space(point $C(1, 1)$) [27]. Similar to AUCROC, the ***area under the PR Curve (AUCPRC)*** is also a summary of PR-Curve with a single scale value [27]. An example of PR-Curve can be observed in Fig 2.5.

Recall, Precision, G-Mean, F1 and AUCPRC are applied to evaluate the algorithms' performance in the following experiment.

Chapter 3

Methodology

In this chapter, the structure and implementation processes of the DDAE model (see section 3.1.1) and the Iterative Metric Learning model (see section 3.1.2) are presented in detail.

3.1 Algorithm Implementation

3.1.1 DDAE

In this section, the process of implementing the novel model DDAE [29] is presented in detail from its four components. **Data Block Construction** is constructed to balance the class distribution. **Data Space Improvement** improves data space for the training set through **Largest Margin Nearest Neighbor(LMNN)** [73]. **Adaptive Weight Adjustment** is inspired by cost-sensitive learning and it generates a weight to make the classifier work better. The last component, **Ensemble Learning**, utilizes major voting and the weight from Adaptive Weight Adjustment component to determine the ultimate prediction. In this algorithm, ***k-nearest neighbors rule*****(kNN)** [7] will be used as the base classifier.

Data Block Construction (DBC) Component

The first step of DBC component is dividing the majority in the training set into several data blocks with a similar size compared to those in the minority in the training set. This can be viewed as an application of undersampling, due to the use of only some of majority samples. Each individually generated data block will be utilized as the training set for each base classifier of the ultimate ensemble classifier. As this approach is introduced to resolve class imbalance problem, the objective of this current component is to achieve better output for each base classifier through balancing the class distribution of each data block. Since one of the significant characteristics of an imbalanced dataset is the huge difference between the class sizes of the majority and minority, DBC applies the concept of oversampling to create more minority samples. This is achieved by copying all the minority samples and placing them in each data block.

Through this procedure, the issue of information loss and overfitting can be improved to some extent,mitigating the arguments against undersampling and oversampling.

The detail of the choice of the number of data blocks $\delta^*(\lceil\delta\rceil$ or $\lfloor\delta\rfloor)$ is explained in Chapter 4.

Algorithm 1 Data Blocks Construction

```
Input: Training Set $D_t$
Output: A set $B$ of $\delta^*$ data blocks
1: Split the majority and minority instances in the training set into $S_{\text{maj}}$ and $S_{\min}$ respectively
   Obtain the size of $S_{\text{maj}}$ and $S_{\min}$, denoted as $N_{\text{maj}}$ and $N_{\min}$
   Obtain the balanced ratio $\delta = N_{\text{maj}}/N_{\min}$
2: Initialize a set of blocks $B = \{B_1, B_2, \ldots, B_{\delta^*}\}$
   ($\delta^*$ can be considered as $\lceil\delta\rceil$ or $\lfloor\delta\rfloor$)
3: Divide the majority samples into $\delta^*$ parts $\{Ma_1, Ma_2, \ldots, Ma_{\delta^*}\}$
4: Copy $S_{\min}$ and $Ma_i$ into one of the empty data blocks $B_i$ which has been created in Step 2.
```

Data Space Improvement (DSI) Component

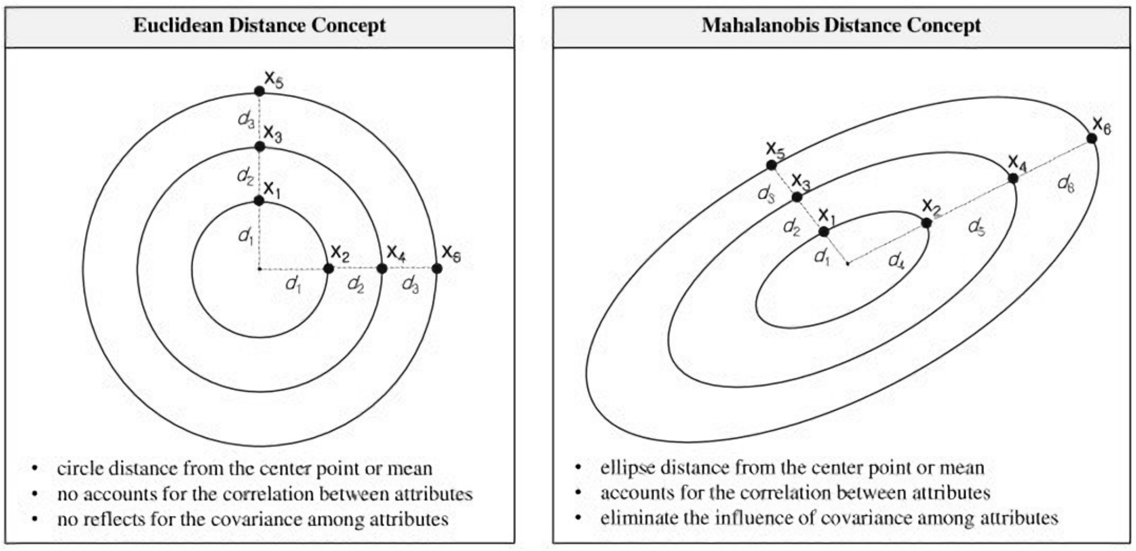

Euclidean Distance Concept
x5
x3
x1
x2
x4
x6
d1
d2
d3
• circle distance from the center point or mean
• no accounts for the correlation between attributes
• no reflects for the covariance among attributes
Mahalanobis Distance Concept
x6
x5
x4
x3
x2
x1
d1
d2
d3
d4
d5
d6
• ellipse distance from the center point or mean
• accounts for the correlation between attributes
• eliminate the influence of covariance among attributes

Figure 3.1: Comparison of Euclidean Distance and Mahalanobis Distance [2, Fig.1]

In this component, a distance metrics learning algorithm called *large margin nearest neighbor(LMNN)* [73] is utilized to improve the data space for training samples in each data block generated in DBC component.

LMNN, first introduced by Kilian Q. Weinberger and Lawrence K. Saul in their 2009 work [73], absorbs the particular advantages of several individual models, such as Mahalanobis Metric for

Clustering (MMC) [74], Pseudometric Online Learning Algorithm (POLA) [74], Neighborhood Component Analysis (NCA) [75]. As the classification approach kNN is sensitive with the metric applied to calculate the distances between different examples, LMNN [73] is proposed to learn ***Mahalanobis distance metrics*** [76] for kNN instead of Euclidean distances used by classical kNN. Figure 3.1 illustrates the difference between these two kinds of distance metric algorithms. The Mahalanobis distance between two data instance points x and y, is defined as

$$^{1}D_M(x,y) = \sqrt{(x-y)^T\Sigma^{-1}(x-y)} \tag{3.1}$$

This proposed approach contains three main parts: a convex loss function, an intention of maximizing margin and the constraints on the distance metric imposed by an accurate kNN algorithm. This, improves the classification accuracy of kNN significantly and it can even be comparable to the SVM classifier on some datasets.

Two simple objectives of LMNN defined in [73] are:

for each training input sample x_i and its label y_i,

1. the label of its k nearest neighbors should be same with y_i
2. the neighbors with different label should be separated widely.

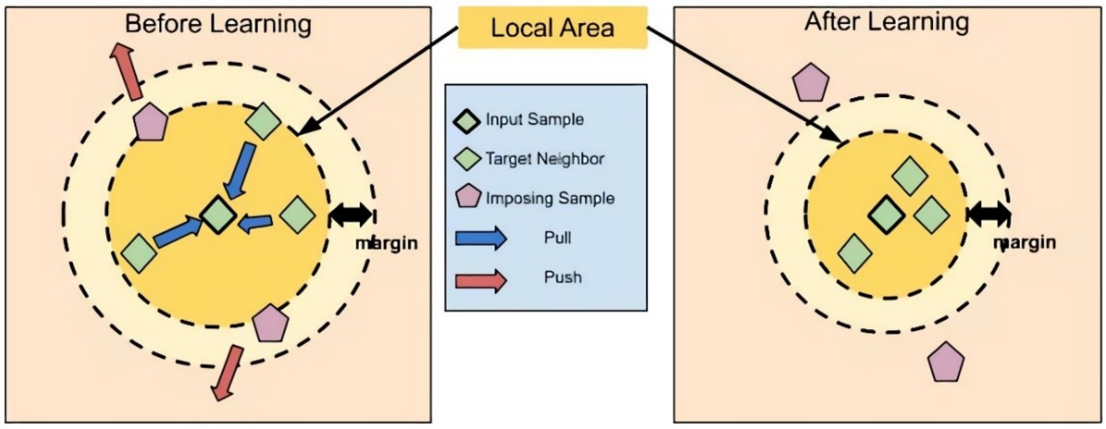


Figure 3.2: The procedure of LMNN Distance Metric Learning

The first stage is, for each data sample x_i, to identify the *target neighbors* that are desired to place close to x_i. In other words, there is specific "margin" around x_i, and other input samples with different labels should be placed further away, so the objective of learning is kind of minimization

[1]Mahalanobis, P. C. (1936). On the generalized distance in statistics. National Institute of Science of India.

of the number of these imposing samples which can be introduced through an inequality:

$$^{2}\left\|L\left(x_i-x_p\right)\right\|^2 \leq\left\|L\left(x_i-x_j\right)\right\|^2+1 \tag{3.2}$$

, with L a linear transformation [73]. For instance, given an input sample x_i with the class label y_i and the target neighbor x_j, the imposing sample x_p should satisfy the inequality. The detail of this procedure with $k=3$ as an example can be observed in Figure 3.2.

Accordingly, the overall loss function should contain two penalizing terms. One of them is used to limit large distances between neighboring input samples with the same class labels, by pulling target neighbors closer to x_i, the other is used to penalize small distances between neighboring input samples with different labels by pushing different labeled samples further away. In [73], the overall loss function for this distance metric learning is a combination of these two mentioned terms, which are denoted as $\varepsilon_{pull}(L)$ and $\varepsilon_{push}(L)$ respectively. With all the previous definitions, it is indicated that these two terms have competing effects, one for attracting the target neighbors and the other for avoiding or repelling the imposing samples. In order to balance these intentions, Weinberger and Saul gives a weighting parameter ω, a positive real number, here. The overall loss function is defined as:

$$^{3}\varepsilon(L)=(1-\omega)\varepsilon_{pull}(L)+\omega\varepsilon_{push}(L) \tag{3.3}$$

, with a general low-dependence on the value of μ, which can be tuned with the help of cross-validation, for the result of minimizing the loss function.

The first term of loss function is defined as

$$^{3}\varepsilon_{pull}(L)=\sum_{j\sim i}\left\|L\left(x_i-x_j\right)\right\|^2 \tag{3.4}$$

, with $j\sim i$ denoted as each x_j, which is the target neighbor of x_i. It should be noted that this term only affects the input samples and their large-distancing target neighbors but not all other input samples with a same class label in the training set.

The second term of the loss function is given as (3.5)

$$^{3}\varepsilon_{\text{push}}(L)=\sum_{i,j\sim i}\sum_{p}\left(1-\theta_{ip}\right)\max\Big(1+\left\|L\left(x_i-x_j\right)\right\|^2-\left\|L\left(x_i-x_p\right)\right\|^2,0\Big) \tag{3.5}$$

, where a new variable y_{ip} is introduced to show whether the x_i and x_p belongs to the identified class. In particular, $\theta_{ip}=1$ if and only if x_i and x_p have the same class label, or $\theta_{ip}=0$. Additionally, the term $\max(m,0)$ refers to the standard hinge loss. To be specific, if the input sample x_p is placed away from x_i safely, this means that its hinge loss is with a negative argument so that there will be

[2]Mahalanobis, P. C. (1936). On the generalized distance in statistics. National Institute of Science of India.

[3]Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research, 10(2).

no effect on the overall loss function.

The last step of this approach is the minimization of this overall loss function. In [73], this can be achieved through gradient descent in the element of L.

Adaptive Weight Adjustment (AWA) Component

Since most ensemble learning approaches assume that the significance of each base classifier (or the weight of each base classifier) is totally equal [77], the purpose of this AWA component is to find an appropriate overall class weight generated using the data coming from each data block [29].

In [29], a definition called ***unstable confusion matrix*** with a similar structure as classical confusion matrix (see Table 2.2) is first introduced in this component, which can be observed in Table 3.1. In detail, the figure for $u(0,0)$(resp. $u(0,1)$) is the number of unstable samples with negative as real label and predicted as negative(resp. positive) and so the figure for $u(1,1)$(resp. $u(1,0)$) represents the number of *unstable samples* with positive as the real label and classified as positive(resp. negative).

	Actual negative	Actual positive
Predict negative	$u(0,0)$ (True Negative)	$u(1,0)$(False Positive)
Predict positive	$u(0,1)$(False Positive)	$u(1,1)$ (True Negative)

Table 3.1: Unstable Confusion Matrix

But how is an *unstable sample* defined here? First, it should be noted that, as the kNN classifier is utilized as a base classifier in this DDAE model, it is probable that the k-nearest neighbors for each data sample are not labeled similarly [29]. One particular instance of this is when the number of two class samples among these k neighbors is not very different. For instance, if k is assumed as 7, and among the 7 nearest neighbors of a data sample s, there are 3 labeled as negative and the other 4 are labeled as positive. According to the kNN rule, these should be predicted as positive but in reality, this decision can be somewhat ambiguous. So this kind of data sample is called an unstable sample, and in [29], the absolute difference between the number of these two class samples, so-called ***positive-negative count difference (PNCD)***, is introduced to distinguish the unstable sample more formally:

$$^4 s \in \begin{cases} \text{unstable samples,} & \text{PNCD } < \rho \\ \text{stable samples,} & \text{otherwise} \end{cases}, \text{ with } \rho = \begin{cases} 1, & k \text{ is odd} \\ 2, & \text{otherwise} \end{cases}. \tag{3.6}$$

To determine PNCD for each test data sample, in each data block, the distances between the specific data sample and each sample in the data block are determined from Euclidean distances.

[4]Yin, J., Gan, C., Zhao, K., Lin, X., Quan, Z., & Wang, Z. J. (2020). A Novel Model for Imbalanced Data Classification. In AAAI (pp. 6680-6687).

After finding k nearest neighbors for each test sample, it is easy to calculate the PNCD for it. According to (3.6), a judgement will be given as to whether this sample belongs to the set of unstable samples or not.

Algorithm 2 Find Neighbors for Test Sample

```
Input: Test Instance $t$, Data Block $b_j$ with $q$ data samples
Output: List $neigh$ with $k$ samples(neighbors)
1: Initialize an empty lists $distance$
2: for $j = 0$ to $q - 1$ do
3:     Calculate the distance $dist_j$ between $b_i[j]$ and $t$
4:     Add $(dist_j, j)$ into list $distance$
5: Sort the list $distance$ according to the first term of each tuple in $distance$
6: for $z = 0$ to $k - 1$ do
7:     Add $b_i$ [distance $[z]$ $[1]$] into $neigh$
8: return $neigh = \{neigh_0, neigh_1, \ldots, neigh_{k-1}\}$
```

As discussed in Chapter 2, the cost of the class positive (minority) is generally higher or more expensive than that of class negative (majority) in imbalanced binary classification. AWA works on the adjustment of weights for the positive and negative predictions and the maximization of the overall gain $g_{overall}$ for the unstable confusion matrix [29]. The weight pair $Weight = (Weight_n, Weight_p)$ consists of the weights for negative predictions and positive predictions. $Weight$ is set to $(Weight_{\text{default}}, Weight_{\text{default}})$ initially with $Weight_{\text{default}} = 1$. If there is an increase with the weights for class positive(resp. negative), g_p (resp. g_n) , g_{overal} is set to g_p (resp. g_n). Otherwise $g_{overall}$ is set to g_d.

$$^{4}\begin{cases} g_d = x * (u(1,1) - u(1,0)) + (u(0,0) - u(0,1)) \\ g_p = x * (u(1,1) + u(1,0) + (-u(0,0) - u(0,1)) \\ g_n = x * (-u(1,1) - u(1,0) + u(0,0) + u(0,1)) \end{cases} \tag{3.7}$$

In order to maximize the $g_{overall}$, the next step is to find the maximal figure among these three gains. The weight pair $Weight = (Weight_{default}, Weight_{default})$ will be updated through a new weight $Weight_{new} = Weight_{threshold} + \gamma$ as follow:

g_d maximal:	$Weight = (Weight_{default}, Weight_{default})$
g_p maximal:	$Weight = (Weight_{default}, Weight_{new})$
g_n maximal:	$Weight = (Weight_{new}, Weight_{default})$

Table 3.2: Updating Weight Pair

[4]Yin, J., Gan, C., Zhao, K., Lin, X., Quan, Z., & Wang, Z. J. (2020). A Novel Model for Imbalanced Data Classification. In AAAI (pp. 6680-6687).

with γ a small real number larger than 0 and $Weight_{threshold}$ defined as (3.8)

$$^{4}Weight_{threshold} = \begin{cases} \frac{\lfloor \frac{\delta^*}{2} \rfloor + 1}{\lfloor \frac{\delta^*}{2} \rfloor - 1}, & \text{if } \delta^* \text{ is odd} \\ \frac{\frac{\delta^*}{2} + 1}{\frac{\delta^*}{2} - 1}, & \text{otherwise} \end{cases} \tag{3.8}$$

Algorithm 3 Determination of overall weight pair

```
Input: threshold $\tau$, $\#default\ weight\ pairs$[5]
       $\#positive\ weight\ pairs$[6], $\#negative\ weight\ pairs$[7]
       $\#all\ weight\ pairs$[8]
Output: Overall weight pair $Weight^{overall}$
1: if $\#default\ weight\ pairs\ /\#all\ weight\ pairs \geq \tau$ then
2:     return $(Weight_{default}, Weight_{default})$
3: else
4:     if $\#positive\ weight\ pairs > \#negative\ weight\ pairs$ then
5:         return $(Weight_{new}, Weight_{default})$
6:     else
7:         return $(Weight_{default}, Weight_{new})$
```

After finishing the above steps, for a single test sample, the result will be $\#data\ block$[9] weight pairs obtained from each data block. The last step(see Algorithm 3) of AWA component is to determine the final overall weight pair $Weight^{overall} = \left(Weight_n^{overall}, Weight_p^{overall}\right)$ for this single test sample, which depends on the frequency of these three different kinds of weight pairs shown in Table 3.2.

Ensemble Learning Component (EL) Component

The main idea behind this component is ensemble learning (see section 2.2.3), with a weighting parameter determined in AWA component. Multiple base classifiers with major voting technique work on the final decision for each input sample. The number of base classifiers m in this ensemble classifier depends on the number of data blocks generated in DBC component, so is $m = \delta^*$.

In particular, this paper focuses on binary classification. At the first stage, for each sample s that needs to be predicted, the number of base classifiers that have predicted it as positive (class label 1) and those as negative (class label 0) need to be counted respectively. This can be denoted as a term

[4]Yin, J., Gan, C., Zhao, K., Lin, X., Quan, Z., & Wang, Z. J. (2020). A Novel Model for Imbalanced Data Classification. In AAAI (pp. 6680-6687).

[5]The number of default weight pairs$(Weight_{default}, Weight_{default})$

[6]The number of non-default positive weight pairs$(Weight_{new}, Weight_{default})$

[7]The number of non-default negative weight pairs$(Weight_{new}, Weight_{default})$

[8]The number of all weight pairs $= 5 + 6 + 7$

[9]The number of data blocks

(n_1, n_0). Namely, for the *i*th base classifier CL_i, the $r_{(i,label)}(s) = 1$ if, and only if, the predicted label for s that made by CL_i is same as $label$, with i in range of 0 to $m-1$ and $label \in \{0,1\}$. With this, it is easy to obtain $n_1 = \sum_{i=0}^{m-1} r_{(i,1)}(s)$ and $n_0 = \sum_{i=0}^{m-1} r_{(i,0)}(s)$.

As introduced in section 2.1.1, the minority or positive class in a class imbalanced dataset always has more impact than the majority or negative class. In order to solve this problem and to enhance the performance of the ensemble classifier, the overall weight $Weight^{overall} = \left(Weight_0^{overall}, Weight_1^{overall}\right)$ is utilized to balance these two classes. Thus, the modified major voting technique can be defined as (3.9).

$$^{10}Res(s) = \begin{cases} label_0, & Weight_0^{overall} * n_0 \geq Weight_1^{overall} * n_1 \\ label_1, & otherwise \end{cases} \tag{3.9}$$

3.1.2 Iterative Metric Learning(IML)

In this section, the process of implementing the model IML [30] is presented in detail with its three components. **Iterative Metric Learning** is constructed to improve the structure of data space through **Largest Margin Nearest Neighbor(LMNN)** [73]. **Iterative Training Samples Selection** finds the samples which are more likely to affect the testing data samples. **Data Matching** determines the most effective neighborhood for each testing data sample. In this algorithm, **k-nearest neighbors rule(kNN)** [7] will be used as the base classifier. Algorithm 4 shows the detail in dealing with imbalanced datasets through IML.

Iterative Metric Learning

In this stage, IML also utilizes the LMNN, which was explained in last section, to improve the data space. The objective of this component is same as that of DSI in DDAE, both of which want to separate the data samples with a different class label by a large margin and make the samples with the same class labels close to each other. Unlike DDAE, IML provides an iterative metric learning approach which can locate a more stable neighborhood for the specific testing data.

Iterative Training Samples Selection

In this stage, after calculating the distance between each of the samples from training set and the to-be-classified testing sample, the top d^p and the top d^f training samples for the positive and negative class will be selected; these chosen samples make up the sub training set for the current testing sample [30].

[10] Yin, J., Gan, C., Zhao, K., Lin, X., Quan, Z., & Wang, Z. J. (2020). A Novel Model for Imbalanced Data Classification. In AAAI (pp. 6680-6687).

Data Matching

After the first two steps, a temporary neighborhood for the current testing sample has already been discovered. However, to find the most effective training set for the testing sample, the first two steps need to be iteratively repeated. In each iteration, the current discovered neighborhood will be compared with the previously discovered neighborhood. If the ratio between the number of matched training samples and the number of all training samples is greater than the matching ratio, this means that the neighborhood is already stable and the current neighborhood is the appropriate training set for this current testing sample, or the iteration needs to continue until the above condition is met.

Algorithm 4 IML

```
Input: Testing set $S^{test}$, Training Set $S^{train}$, matched ratio $\beta$, kNN Classifier $K$
Output: List of predictions for the given testing set $P$
 1: for $s^{test}$ in $S^{test}$ do
 2:     Initialize $Pre\_set$ and $Curr\_Set$
 3:     while true do
 4:         do Metric Learning on $S^{train}$ and update $S^{train}$
 5:         do Training Samples Selection and update the $Curr\_Set$
 6:         do Data Matching and find the matched samples
 7:         if #matched samples / size($Curr\_Set$) $>= \beta$ then
 8:             trained the $K$ using $Curr\_Set$, add the prediction into $P$
 9:             break
10:         else
11:             $Pre\_set = Curr\_Set$
12: end for
13: return $P$
```

Chapter 4

Experiments and Results

This chapter reports the results of the experiments on imbalanced medical/healthcare datasets in detail. Section 4.1 introduces the datasets used as input. This is followed by section 4.2, which reports the results of all the experiments. Section 4.2.1 presents an overall comparison of all class imbalance classification models. Section 4.2.2 and section 4.2.3 illustrates the result of whether a different imbalanced ratio and the different size of the data set have an impact on the performance of the model. Moreover, in section 4.3, the effectiveness of all components in the DDAE is described.

4.1 Input-Data Used

4.1.1 Source of Datasets

Eight datasets were collected from the medical or healthcare sector. These are Yeast1vs7, Euthyroid Sick, Thyroid Sick, and Mammographic (MGC), Wisconsin Diagnosis Breast Cancer(WDBC) and Pima Indian Diabetes(PID) from UCI [78], and two sub datasets of Protein Homology (PH1 and PH2) from KDD Cup 2004 [79], which are utilized to test the performance of these models. The detail of these datasets is depicted in Table 4.1. In addition, eight further datasets are employed, including Cm1, Mw1, Pc1, Pc3, Pc4 which are from NASA Metrics Data Program (NASA) dataset [80], Poker89vs6 and Poker8vs6, which are from KEEL [81], and Optical Recognition of Handwritten Digits (optdigits) from UCI. All these datasets are imbalanced distributed but with various imbalance ratio(IR), instances and features. The detail of these datasets is depicted in Table 4.2.

Euthyroid Sick and Thyroid Sick record the patient information of the thyroid disease. In Mammographic (MGC), according to the attributes of BI-RADS and the age of patients, breast and benign breast masses are distinguished. Wisconsin Diagnosis Breast Cancer(WDBC) records the data about diagnostic Wisconsin breast cancer. Pima Indian Diabetes(PID) records the patient information of diabetes disease, and all patients here are females at least 21 years old of Pima Indian heritage.

Protein Homology dataset consists of the recording about protein homology. Yeast1vs7 contains the data about the cellular localization sites of proteins. Poker8v6 and Poker89vs6 are used for poker hands prediction. In addition, optdigits contains normalized bitmaps of handwritten digits from a preprinted form.

Dataset	#Class	#Instances	#F	IR
Euthyroid Sick	2	3,163	42	9.795
Thyroid Sick	2	3,772	52	15.329
PH1	2	11,274	74	7.699
PH2	2	31,296	74	23.148
MGC	2	11,183	6	42.012
Yeast1vs7	2	459	7	14.3
WDBC	2	768	8	1.866
PID	2	568	32	1.684

Table 4.1: Characteristics of used Datasets from Medical or Healthcare Sector

Dataset	#Class	#Instances	#F	IR
optdigits	2	5,620	65	9.144
Cm1	2	497	21	9.354
Mw1	2	403	37	12
Pc1	2	1109	21	13.4
Pc3	2	1563	37	8.769
Pc4	2	1458	37	7.191
Poker89vs6	2	1485	10	58.4
Poker8vs6	2	1477	10	85.882

Table 4.2: Characteristics of used Datasets from Other Fields

4.1.2 Data Pre-processing

As described in section 2.1, the datasets in the real world are generally not as perfect as expected. They can be unreliable or dirty, with lack of attribute values, containing noise (errors or outliers) and inconsistent or duplicate values [82]. The performance of classifiers is dependent on the quality of training data. A dirty or incorrect training set can result in a poor classification model [83]. In order to reduce the impact of these issues on prediction models, data pre-processing, a critical part of data analysis and machine learning, takes a huge amount of time [84]. In this paper, the experiment focuses on the performance of the model, so here only a brief pre-processing is performed on the input datasets.

In [84], the process of data preparation is depicted in four steps: data integration, data cleaning, data normalization and data transformation. However in other literature, such as [85], this process is presented more detail and data normalization is seen as a part of data transformation.

When the data is obtained from multiple data sources, step one is critical wherein all the data collected needs to be integrated [84]. This procedure involves the identification of redundant attributes, in which the size of dataset can be decreased, also reducing the modeling time of further algorithms [84], as well as the detection of duplication and inconsistency. As some identical instances are seen as different due to errors in the entry step, duplication is a cause of inconsistency which means detection is indispensable [86].

Data cleaning is performed to deal with dirty data [84], which includes missing values, incorrect data and data with non-standard representation. A high proportion of this kind of data lead to an ineffective and unreliable algorithm [83]. This procedure can be divided into three parts: missing value processing, outliers processing and noise processing [85].

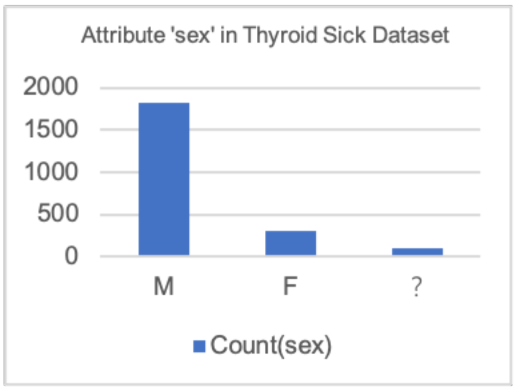


Figure 4.1: Distribution of Attribute 'sex' in Thyroid Sick Dataset

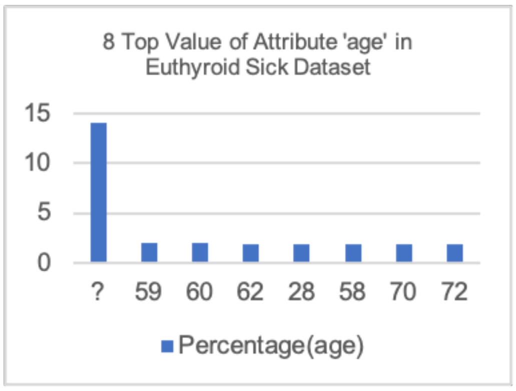


Figure 4.2: 8 Top Values of Attribute 'age' in Euthyroid Sick Dataset

In the real world, during the process of acquiring information and data, there will be various causes of data loss and vacancies. There are several options for filling in the missing value. If the missing rate of an attribute is high, this attribute can be deleted directly, so-called listwise deletion [87]. Alternatively, the missing values can be set to specific values calculated on the training set, such as zero, the mean, the median, etc. [88]. For instance, in the Euthyroid Sick Dataset, the missing rate of the attribute 'Age' is 14.10% (466 of 3163). Figure 4.2 shows the 8 top values of this attribute, with the age range of the whole dataset between 1 and 98 years old. During the data cleaning process, this kind of missing value will be replaced with the median of this attribute. Dummy variable adjustment [89] is utilized when the attribute is discrete and has few different values, which can be converted into a dummy variable. For example, in the Thyroid Sick dataset, there are three different values for the gender SEX attribute: 'M[11]', 'F[12]', '?[13]' (see Figure 4.1, so this column can be converted into IS_SEX_MALE, IS_SEX_FEMALE, IS_SEX_NA.

Another kind of data that needs to be cleaned is outlier. Outliers are data points that differ from the

[11] Male
[12] Female
[13] Missing Value

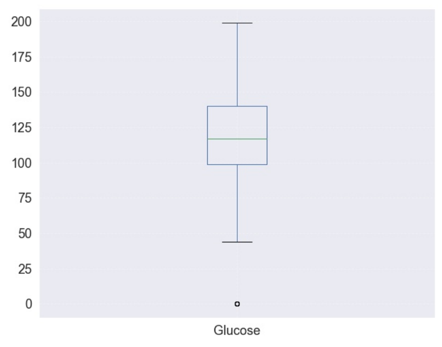


Figure 4.3: Box Plot for Attribute 'Glucose' in PID Dataset

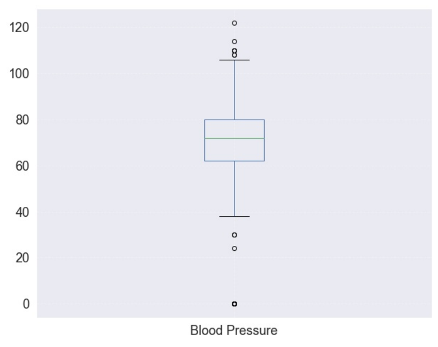


Figure 4.4: Box Plot for Attribute 'Blood Pressure' in PID Dataset

rest of the data due to human error, mechanical faults, instrument errors etc. [90], which is viewed as a normal state of data distribution, and this kind of data outside a specific distribution area or range is also defined as anomalies or abnormalities [91]. One of the visualization methods applied to detect this is box plot using quantiles. Figure 4.3 and Figure 4.4 depict the box plots for the attributes 'Blood Pressure' and 'Glucose' from the PID dataset. The outliers can be observed clearly in this illustration. There are some instances with an extremely low value for 'Blood Pressure' which is rare in the real world. An instance of a 0 'Glucose' can also be viewed as an outlier. Such incorrect data, or data that violates common sense, may lead to an ineffective model.

There may be variation in the scales of different data features and in the differences between the values [88]. Note the case in the WDBC dataset: the attribute 'area_mean' ranges from 143.5 to 2,501 but the attribute 'smoothness_mean' only ranges from 0.053 to 0.163. This phenomenon may have an impact on the results of data analysis, especially the distance-based algorithms, such as SVM. The data must also be measured to a certain ratio such that they fall into a particular field for detailed analysis [84]. Three normalization procedures [84, 88, 92] utilized for an attribute a generally are:

Min-Max Normalization[14] $$v' = \frac{v - \min_a}{\max_a - \min_a}(new_{\max_a} - new_{\min_a}) + new_{\min_a}$$

Z-score Normalization[15] $$v' = \frac{v - mean_a}{stand\_dev_a}$$

Decimal Scaling Normalization[16] $$v' = \frac{v}{10^j}$$

[14] All the numerical values of the specific numerical attribute a are scaled to a specific range $[new_{\min_a}, new_{\max_a}]$

[15] $mean_a$ is the sample mean and $stand\_dev_a$ is the sample standard deviation, which is computed as $\frac{1}{n}\sum_{i=1}^{n}|v_i - mean_a|$

[16] where j is the smallest integer such that $new_{\max_a} < 1$

Raw Number	referral source
1	SVI
2	STMW
3	SVHD
4	SVHC
5	other

$\Longrightarrow$

RSource1	RSource2	RSource3	RSource4
1	0	0	0
0	0	1	0
0	0	0	1
0	1	0	0
0	0	0	0

Table 4.3: Comparison of Before and After One Hot Encoding for Attribute 'referral source' in Thyroid Sick Dataset

Other useful steps are feature encoding [88], feature selection [93], dimensionality reduction [88] etc. Since most models have a preference for processing numerical data, handling some text and categorical attributes, feature encoding is a kind of useful data transformation to make data more acceptable as input for models [88]. The attribute 'referral source' of thyroid sick dataset contains five values: 'SVI', 'SVHC', 'STMW', 'SVHD' and 'other'. Table 4.3 shows an example of this attribute before and after transformation. This attribute will be replaced by four new attributes: 'RSource1', 'RSource2', 'RSource3' and 'RSource4'. This is also an application of dummy variables adjustment and those new attributes are called dummy attributes. The package Scikit-Learn provides a OneHotEncoder class to transform these kinds of category values into one-hot vectors [88].

There is no set series of specific data pre-processing methods that can be applied to all data sets, so for different datasets, specific analysis needs to be carried out according to its characteristics [86].

4.1.3 Split of Dataset into Training Set and Testing Set

For evaluating machine learning algorithms, the Train-Test Split is indispensable, which involves the division of a given dataset into two subset according to a specific ratio. The subset applied to fit the algorithm is called the training set, and the other is referred to as the testing set and is utilized to evaluate the fit algorithm [94]. To be exact, available data with known input vector and target values works on fitting the given model, so that it can predict new instances without class values. In this paper, all the given datasets will be split based on Train:Test = 7:3.

4.2 Results on All Models

4.2.1 Overall Comparison

Tables 4.4-4.13 are provided for each separate dataset, listing the performance of each of the models applied in terms of G-mean, F1 and AUCPRC.

Models	Euthyroid Sick		
	G-mean	**F1**	**AUCPRC**
DDAE	0.880	0.552	0.587
MWMOTE	0.832	0.658	0.616
SMOTE	0.829	0.667	0.621
RUSBoost	0.911	0.725	0.779
AdaBoost	0.904	**0.862**	**0.901**
MetaCost	0.935	0.846	0.723
csDCT	0.960	0.846	0.723
CAdaMEC	0.876	0.831	0.865
self-paced	**0.971**	0.856	0.861
IML	0.867	0.809	0.836
Average	0.897	0.765	0.751

Table 4.4: Overall Results for Euthyroid Sick Dataset

Models	Thyroid Sick		
	G-mean	**F1**	**AUCPRC**
DDAE	0.883	0.506	0.622
MWMOTE	0.735	0.558	0.342
SMOTE	0.694	0.535	0.323
RUSBoost	0.908	0.783	0.805
AdaBoost	0.879	0.828	0.882
MetaCost	0.413	0.280	0.342
csDCT	0.891	0.811	0.700
CAdaMEC	0.835	0.794	**0.901**
self-paced	**0.915**	**0.861**	0.893
IML	0.615	0.474	0.335
Average	0.777	0.643	0.615

Table 4.5: Overall Results for Thyroid Sick Dataset

Models	PH1		
	G-mean	**F1**	**AUCPRC**
DDAE	**0.955**	0.828	0.891
MWMOTE	0.934	0.781	0.647
SMOTE	0.925	0.765	0.630
RUSBoost	0.938	0.868	0.940
AdaBoost	0.940	**0.926**	**0.963**
MetaCost	0.851	0.774	0.706
csDCT	0.918	0.837	0.728
CAdaMEC	0.934	0.913	0.960
self-paced	0.936	0.892	0.943
IML	0.907	0.874	0.925
Average	0.924	0.846	0.833

Table 4.6: Overall Results for PH1 Dataset

Models	**PH2**		
	G-mean	**F1**	**AUCPRC**
DDAE	0.986	0.830	0.956
MWMOTE	0.953	0.664	0.498
SMOTE	0.935	0.701	0.546
RUSBoost	0.996	0.981	0.999
AdaBoost	0.995	0.994	**1**
MetaCost	0.884	0.794	0.783
csDCT	**0.999**	**0.999**	0.997
CAdaMEC	0.995	0.991	**1**
self-paced	**0.999**	**0.999**	**1**
IML	0.978	0.964	0.999
Average	0.972	0.892	0.878

Table 4.7: Overall Results for PH2 Dataset

Models	**optdigits**		
	G-mean	**F1**	**AUCPRC**
DDAE	**0.986**	0.904	**0.995**
MWMOTE	**0.986**	0.965	0.934
SMOTE	0.983	0.968	0.940
RUSBoost	0.970	0.921	0.971
AdaBoost	0.963	0.955	0.986
MetaCost	0.836	0.616	0.490
csDCT	0.900	0.758	0.637
CAdaMEC	0.972	**0.985**	0.992
self-paced	0.965	0.946	0.980
IML	0.981	0.970	**0.995**
Average	0.954	0.899	0.892

Table 4.8: Overall Results for optdigits Dataset

Models	**MGC**		
	G-mean	**F1**	**AUCPRC**
DDAE	**0.878**	0.321	0.324
MWMOTE	0.798	0.522	0.321
SMOTE	0.813	0.539	0.331
RUSBoost	0.708	0.549	0.513
AdaBoost	0.803	**0.767**	**0.758**
MetaCost	0.793	0.313	0.208
csDCT	0.782	0.284	0.141
CAdaMEC	0.640	0.562	0.627
self-paced	0.877	0.516	0.669
IML	0.411	0.286	0.270
Average	0.750	0.466	0.416

Table 4.9: Overall Results for MGC Dataset

Models	WDBC		
	G-mean	**F1**	**AUCPRC**
DDAE	0.866	0.831	0.825
MWMOTE	0.891	0.873	**0.957**
SMOTE	**0.941**	**0.937**	0.945
RUSBoost	0.911	0.898	0.926
AdaBoost	0.887	0.871	0.832
MetaCost	0.887	0.871	0.910
csDCT	0.907	0.882	0.831
CAdaMEC	0.891	0.873	**0.957**
self-paced	0.866	0.831	0.825
IML	0.893	0.870	0.910
Average	0.894	0.874	0.892

Table 4.10: Overall Results for WDBC Dataset

Models	PID		
	G-mean	**F1**	**AUCPRC**
DDAE	0.712	0.657	0.586
MWMOTE	0.760	0.780	0.841
SMOTE	**0.807**	**0.824**	**0.859**
RUSBoost	0.649	0.562	0.642
AdaBoost	0.693	0.671	0.709
MetaCost	0.751	0.688	0.734
csDCT	0.738	0.671	0.610
CAdaMEC	0.727	0.658	0.692
self-paced	0.640	0.550	0.610
IML	0.667	0.587	0.586
Average	0.714	0.665	0.687

Table 4.11: Overall Results for PID Dataset

Models	Yeast1vs7		
	G-mean	**F1**	**AUCPRC**
DDAE	0.713	0.214	0.257
MWMOTE	**0.778**	0.307	0.336
SMOTE	0.775	0.300	**0.468**
RUSBoost	0.598	0.353	0.304
AdaBoost	0.496	0.333	0.305
MetaCost	0.331	0.080	0.083
csDCT	0	0	0.129
CAdaMEC	0	0	0.066
self-paced	0.740	0.245	0.181
IML	0.607	**0.526**	0.387
Average	0.582	0.237	0.304

Table 4.12: Overall Results for Yeast1vs7 Dataset

Models	**Poker8vs6**		
	G-mean	**F1**	**AUCPRC**
DDAE	0.622	0.029	0.078
MWMOTE	0.866	0.857	0752
SMOTE	0.999	0.889	0.800
RUSBoost	0.707	0.667	0.761
AdaBoost	**1**	**1**	**1**
MetaCost	0	0	0
csDCT	0.443	0.019	0.009
CAdaMEC	0	0	0.007
self-paced	0.613	0.085	0.038
IML	0.500	0.400	0.379
Average	0.695	0.523	0.482

Table 4.13: Overall Results for Poker8vs6 Dataset

Models	**Poker89vs6**		
	G-mean	**F1**	**AUCPRC**
DDAE	0.704	0.051	0.188
MWMOTE	0.986	0.500	0.750
SMOTE	0.979	0.400	0.750
RUSBoost	**1**	**1**	**1**
AdaBoost	**1**	**1**	**1**
MetaCost	0	0	0.015
csDCT	0.495	0.032	0.019
CAdaMEC	0	0	0.014
self-paced	0.567	0.036	0.016
IML	0.816	0.800	0.848
Average	0.755	0.482	0.559

Table 4.14: Overall Results for Poker89vs6 Dataset

Models	**Cm1**		
	G-mean	**F1**	**AUCPRC**
DDAE	0.699	0.29	0.214
MWMOTE	0.679	0.286	0.182
SMOTE	0.715	0.308	0.219
RUSBoost	0.438	0.25	0.386
AdaBoost	**0.981**	**0.970**	**0.995**
MetaCost	0.744	0.322	0.198
csDCT	0.559	0.203	0.158
CAdaMEC	0.622	0.237	0.196
self-paced	0.690	0.298	0.262
IML	0.499	0.207	0.137
Average	0.663	0.337	0.295

Table 4.15: Overall Results for Cm1 Dataset

Models	**Pc1**		
	G-mean	**F1**	**AUCPRC**
DDAE	**0.740**	0.237	0.179
MWMOTE	0.718	0.226	0.160
SMOTE	0.734	0.244	0.205
RUSBoost	0.440	0.235	0.214
AdaBoost	0.223	0.091	0.226
MetaCost	0.548	0.197	0.109
csDCT	0.642	0.185	0.099
CAdaMEC	0.673	0.203	0.156
self-paced	0.649	**0.253**	**0.251**
IML	0.649	**0.253**	**0.251**
Average	0.602	**0.212**	0.185

Table 4.16: Overall Results for Pc1 Dataset

Models	**Pc4**		
	G-mean	**F1**	**AUCPRC**
DDAE	0.756	0.381	0.364
MWMOTE	0.825	0.497	0.499
SMOTE	0.789	0.460	0.443
RUSBoost	0.752	0.418	0.413
AdaBoost	0.823	**0.699**	**0.714**
MetaCost	0.790	0.436	0.273
csDCT	0.826	0.512	0.364
CAdaMEC	0.831	0.475	0.695
self-paced	**0.837**	0.524	0.453
IML	0.668	0.505	0.416
Average	0.790	0.491	0.463

Table 4.17: Overall Results for Pc4 Dataset

Models	**Pc3**		
	G-mean	**F1**	**AUCPRC**
DDAE	0.772	0.440	0.319
MWMOTE	**0.780**	**0.491**	**0.401**
SMOTE	0.752	0.440	0.376
RUSBoost	0.452	0.273	0.310
AdaBoost	0.438	0.289	0.343
MetaCost	0.737	0.432	0.263
csDCT	0.719	0.432	0.284
CAdaMEC	0.629	0.301	0.336
self-paced	0.719	0.432	0.284
IML	0.418	0.267	0.300
Average	0.645	0.380	0.322

Table 4.18: Overall Results for Pc3 Dataset

What can be observed from these results is that the G-mean of DDAE is close to or better than the average on most of the datasets. For example, the G-mean value obtained on the MGC is 0.878, while the average value at this time is only 0.750; the G-mean value yielded on Pc1 is 0.740, while the figure for RUSBoost is only 0.44. However, it should be noted that DDAE's F1 and AUCPRC are not satisfactory; they are, lower than the corresponding average values on most of the datasets. In comparison, the two ensemble algorithms, AdaBoost and self-paced Ensemble Classifier, can maintain satisfactory F1, AUCPRC under most circumstances, as well as G-mean,showing that these two algorithms can accurately classify not only a large proportion of minority but also most majority in most cases. For example, AdaBoost has achieved the highest F1 and AUCPRC on datasets such as Euthyroid Sick and PH1, its F1(0.970) being nearly three times that of MetaCost(0.322), which is the second highest value, on the Cm1 dataset; self-paced Ensemble Classifier has achieved the highest F1 and AUCPRC on Thyroid Sick dataset and PH2 dataset, and its F1 and AUCPRC on Pc1 are twice that of AdaBoost. In addition, the performance of another ensemble algorithm, RUSBoost, is relatively stable, with performance on most datasets being close to the average.

The stability of cost-sensitive learning models, such as MetaCost, csDCT and CAdaMEC, is not very obvious. CAdaMEC, as well as csDCt, performed well on the optdigits dataset and the Thyroid Sick dataset, but on Yeast1vs7 dataset, its performance is the worst.

SMOTE and MWMOTE are two sampling models, with a similar performance in most cases except on the Poker8vs6 dataset.

Moreover, the performance of IML fluctuates above and below the average level.Compared with other algorithms, the advantage is not obvious.

Dataset	**Recall**									
	DDAE	MWMOTE	SMOTE	RUSBoost	AdaBoost	MetaCost	csDCT	CAdaMEC	self-paced	IML
Euthyroid Sick	**0.929**	0.737	0.725	0.888	0.827	0.898	0.867	0.776	0.847	0.763
Thyroid Sick	**0.883**	0.558	0.494	0.844	0.779	0.169	0.805	0.701	0.844	0.383
PH1	**0.956**	0.926	0.913	0.900	0.887	0.741	0.867	0.877	0.887	0.831
PH2	0.990	0.946	0.939	0.992	0.990	0.787	**1**	0.990	0.997	0.931
MGC	**0.834**	0.651	0.675	0.506	0.675	0.675	0.663	0.410	0.795	0.169
WDBC	0.894	0.950	**0.960**	0.864	0.894	0.818	0.909	0.833	0.864	0.864
PID	0.810	0.824	**0.869**	0.512	0.548	0.774	0.690	0.631	0.524	0.524
Yeast1vs7	**0.750**	**0.750**	**0.750**	0.375	0.250	0.125	0	**0.750**	**0.750**	0.375

Table 4.19: Recall for for All Models on the Nine Medical Public Datasets

In order to investigate the ability of theses models mentioned in this paper on positive sample detection, which is important for the field of medical or healthcare, the recall for all these models are presented in Table 4.19.

It can be observed that the recall for DDAE performs well on all medical datasets. The recall for self-paced Ensemble Classifier and CAdaMEC can also maintain a relatively stable performance on nearly all the medical datasets.

Dataset	Learning Time(in Seconds)									
	DDAE	MWMOTE	SMOTE	RUSBoost	AdaBoost	MetaCost	csDCT	CAdaMEC	self-paced	IML
Euthyroid Sick	146.775	0.381	0.224	0.050	0.110	2.426	7.586	0.185	**0.041**	912.874
Thyroid Sick	471.882	1.362	1.711	0.542	1.103	19.91	16.786	0.370	**0.08**	1680.46
PH1	1753.42	0.940	8.006	4.857	2.494	64.927	786.164	1.053	**0.226**	16627.12
PH2	14446.34	115.313	51.917	4.091	5.783	188.748	4513.89	2.770	**0.302**	46155.96
optdigits	637.385	0.983	3.207	0.863	0.976	31.625	166.284	0.372	**0.07**	5001.329
MGC	1472.36	0.845	0.749	0.529	0.679	54.577	31.973	0.231	**0.065**	9861.933
WDBC	13.426	0.950	0.960	**0.864**	0.894	0.818	0.909	16.117	**0.864**	8.106
PID	17.45	0.381	0.224	0.381	0.478	9.187	15.547	0.089	**0.029**	27.034
Yeast1vs7	46.994	0.083	**0.014**	0.235	0.141	2.411	11.044	0.109	0.036	5.85
Cm1	54.638	0.069	0.060	**0.016**	0.106	1.201	36.415	0.086	0.029	10.873
Pc1	76.463	0.103	0.119	0.244	0.177	3.57	35.073	0.106	**0.032**	8.702
Pc3	110.98	0.358	0.053	0.274	0.258	5.388	136.251	0.157	**0.034**	180.428
Pc4	97.79	0.399	0.638	0.259	0.208	5.175	65.156	0.118	**0.038**	156.004
Poker8vs6	126.932	0.101	**0.034**	0.185	0.135	4.565	7.267	0.090	0.038	134.869
Poker89vs6	101.163	0.136	**0.020**	0.236	0.261	4.617	6.322	0.102	0.029	135.836

Table 4.20: Learning Time for All Models on the Public Datasets

Table 4.20 shows the learning time achieved by each algorithms on different datasets. It can be seen that IML and DDAE yields the two most longest learning time compared to the other algorithms. Self-paced Ensemble Classifier performs fairly well in terms of learning time with most of the values ranging in (0, 0.1). Considering the characteristics of the datasets described in section 4.1.1, it can be illustrated that, the larger the dataset is, the longer the learning time required for the algorithms.

4.2.2 Impact of Imbalanced Ratio

First, to analyze the influence of IR on the model, eight identically-sized sub-datasets with different imbalance ratios(IRs) were used in the experiment. They were IR=10, 20, 30, 40, 50, 60, 70, 80 and 90. The x-axis for Figures 4.5-4.7 is IR.

Figure 4.5 illustrates the trend of recall of all algorithms as the IR increases. Clearly, this metric stays stable on DDAE most of the time, and DDAE keeps the recall between 0.9 and 0.95. MWMOTE, SMOTE, cost-sensitive Decision Tree, CAdaMEC, self-paced Ensemble Classifier and IML show a downward trend. Among these, the recall of MWMOTE, SMOTE and self-paced Ensemble Classifier decreased slightly as the class distribution became more imbalanced, with the highest and lowest values for MWMOTE, SMOTE, AdaBoost and self-paced Ensemble Classifier being 0.978(IR=30) and 0.852 (IR=90), 0.967 (IR=30) and 0.793 (IR=80), 0.865 (IR=10) and 0.517 (IR=80), 0.898 (IR=10) and 0.795 (IR=80), respectively. The figures for recall of the other four "decreasing" algorithms drop significantly, from 0.841(IR=20) to 0.593(IR=90) for csDCT, from 0.836(IR=10) to 0.519(IR=90) for CAdaMEC, and from 0.767(IR=10) to 0.517(IR=90) for IML. In addition, no obvious correlation between recall and the changes in IR can be seen through the curves of RUSBoost and MetaCost. All of them show a slight fluctuation in this process.

Notably, the G-mean of almost all the models remain nearly stable with the exception of MetaCost. All of the models still retain a high value for G-mean even when the IR is 70 or 80. It can be observed from Figure 4.6 that some of models performs better (or similarly) when the IR is 70 compared to the IR is 10, including DDAE, MWMOTE, SMOTE, RUSBoost, AdaBoost, MetaCost, self-paced Ensemble Classifier, IML and csDCT.

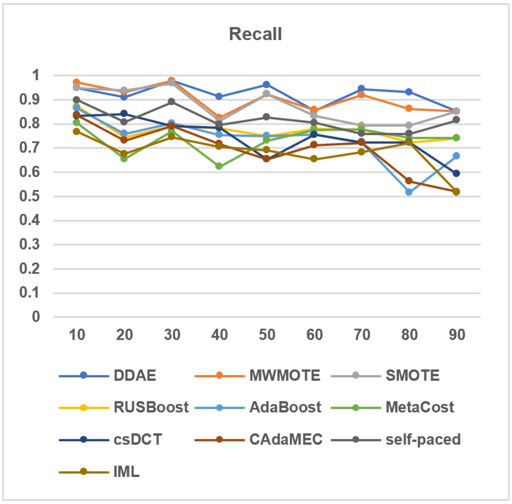


Figure 4.5: Recall: Impact of Imbalance Ratio

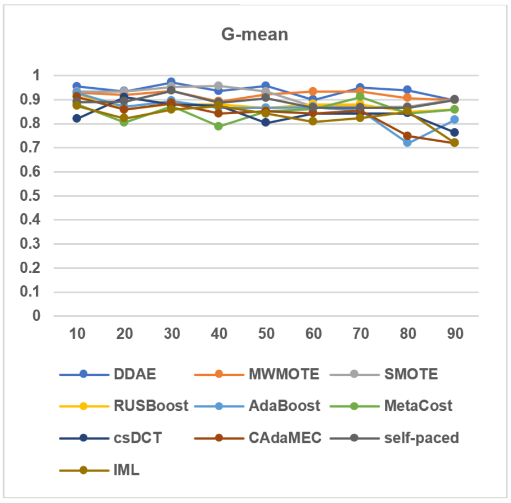


Figure 4.6: G-mean: Impact of Imbalance Ratio

F1 is the evaluation metric which takes both recall and precision into account, which means that if the trend of recall for one specific model maintain stability with only a slight decrease/increase, the changes in F1 will be similar to those in precision. According to the above description of the changes in recall, the trends for F1 of SMOTE, MWMOTE, DDAE, csDCT can be compared with those for their precision, which can be observed from Figure 4.7. The value of F1 for MetaCost and AdaBoost decrease slightly during the process. In contrast, as the performance of other models shows degradation to some degree, the figures for IML, RUSBoost and self-paced Ensemble Classifier stay almost stable, but the former increases slightly after IR is greater than 70 and the latter drops to a small degree. AUCPRC is another evaluation metric which considers recall and precision at the same time. According to Figure 4.8, the AUCPRCs for DDAE, MWMOTE, SMOTE, CAdaMEC and IML decrease as the IR increases. The figure for MetaCost fluctuates significantly but is still showing a decrease. The AUCPRCs for AdaBoost, RUSBoost and self-paced Ensemble

Classifier are all on a slightly downward trend. Moreover, csDCT preforms differently on AUCPRC from other models, the figure for these two models climbing when the IR varies from 50 to 90.

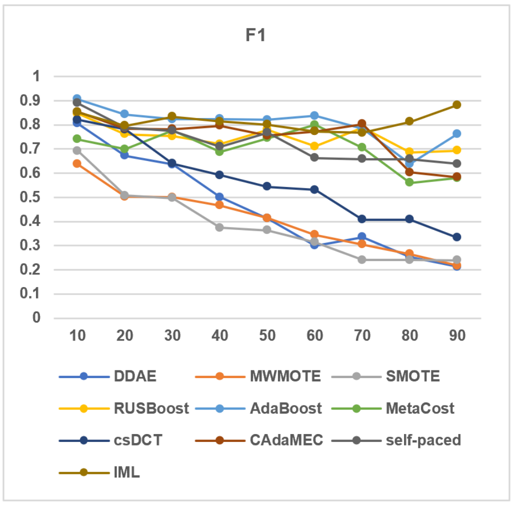


Figure 4.7: F1: Impact of Imbalance Ratio

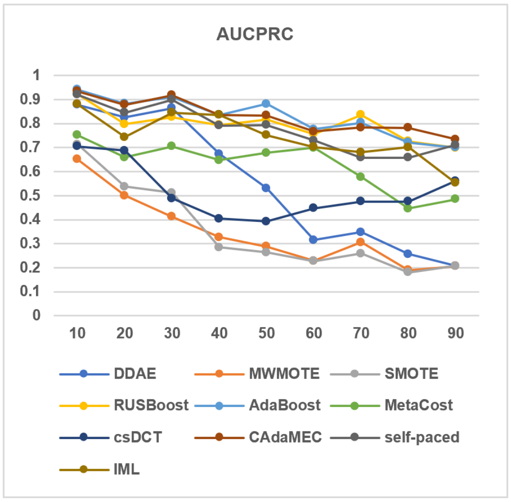


Figure 4.8: AUCPRC: Impact of Imbalance Ratio

4.2.3 Impact of the Size of Datasets

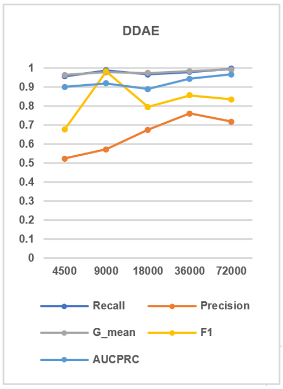


Figure 4.9: DDAE: Impact of the Size of Dataset

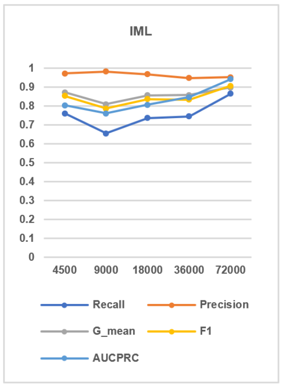


Figure 4.10: IML: Impact of the Size of Dataset

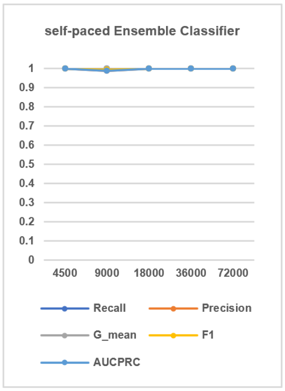


Figure 4.11: self-paced: Impact of the Size of Dataset

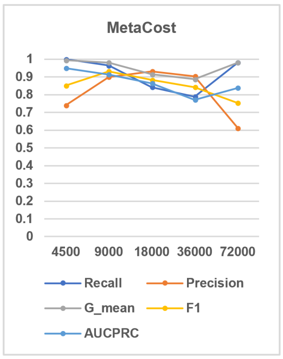


Figure 4.12: MetaCost: Impact of the Size of Dataset

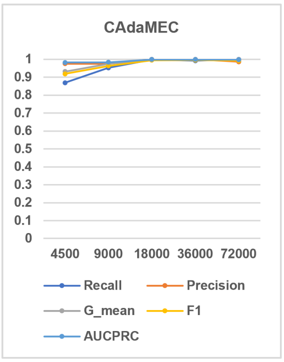


Figure 4.13: CAdaMEC: Impact of the Size of Dataset

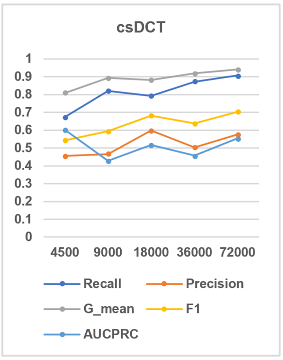


Figure 4.14: csDCT: Impact of the Size of Dataset

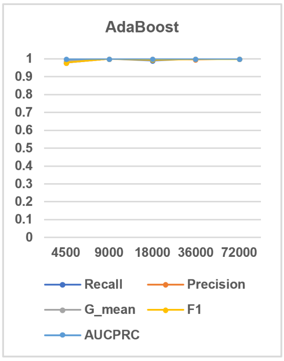


Figure 4.15: AdaBoost: Impact of the Size of Dataset

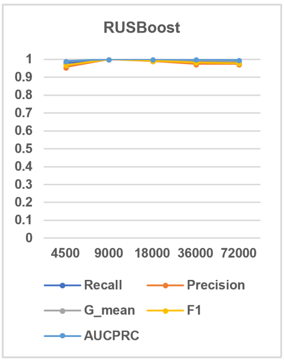


Figure 4.16: RUSBoost: Impact of the Size of Dataset

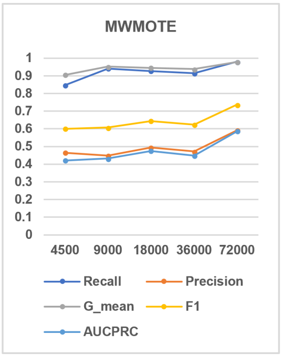


Figure 4.17: MWMOTE: Impact of the Size of Dataset

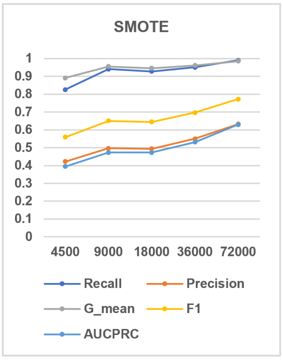


Figure 4.18: SMOTE: Impact of the Size of Dataset

Next, in order to analyze whether the size of the dataset will affect the classification accuracy of the model, five sub-datasets with the same IR but a different number of instances (which are #Instances=4500, 9000, 18000, 36000 and 72000) are taken from the Protein Homology dataset. The results can be observed in Figures 4.9-4.18.

It can be seen from the figures that in the process of increasing the sample size of the algorithms DDAE, MWMOTE, SMOTE, IML and cost-sensitive Decision Tree(csDCT), the values of the five evaluation metrics have shown an upward trend, especially the precision and F1, which increases more than 0.2 and 0.1 respectively. This phenomenon shows that the classification results are more accurate on these models when the sample size is extensive compared to when the sample size is small.

The RUSBoost, AdaBoost, self-paced Ensemble Classifier and CAdaMEC algorithms are not significantly affected by the size of the dataset. The performance of the algorithms on all five experiments are quite excellent, and the value of the evaluation metrics are mostly between 0.9-1.0. Although the CAdaMEC is slightly inadequate when the total sample size is 4500, the recall value is also greater than 0.8, and as the total number of instances exceeds 10,000, this model can predict the labels of all majority and minority class correctly among all the algorithms.

It can be noted that, only MetaCost does not show an apparent trend.

4.3 Results on Effectiveness of DDAE

4.3.1 Results on Effectiveness of Components

The DDAE model includes a total of four components, namely: Data Block Construction(DBC), Data Space Improvement(DSI), Adaptive Weight Adjustment(AWA) and Ensemble Learning(EL). In order to analyze the effectiveness of each component in the entire model, DDAE is modified by deleting the components except for EL in the model. After that, three variants are created: DDAE-DBC, DDAE-AWA and DDAE-DSI. It should be noted that since DDAE itself is an ensemble model, EL is indispensable at all times, which is why the DDAE-El is not listed in these variants. It can be observed from Figure 4.18 and Figure 4.19 that on two datasets(PH1 and Euthyroid Sick) when the model removes the DBC module, the recall and G-mean drops significantly, which is the poorest performance among all variants and the original model. At the same time, it can be seen that the precision varies distinctly across the variants. But the recall of DDAE without AWA or DSI does not drop as much. The performance of the original model without DSI also can be affected, which can be observed from the Figure 4.18 and Figure4.19 for all evaluation metrics. Without the improvement to data space, the accuracy of prediction may decrease. All in all, the integrated algorithm of DDAE absorbs the advantages of each component so that the minority classes can be correctly classified.

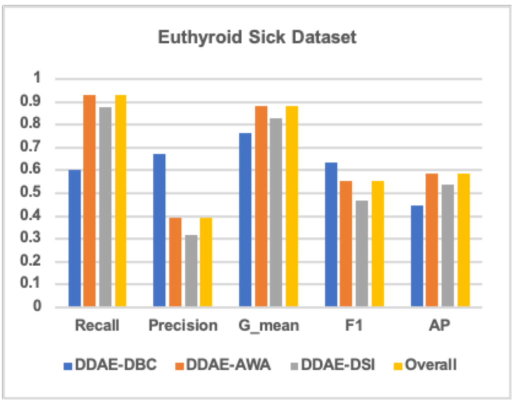


Figure 4.19: Effectiveness of DDAE on Euthyroid Sick Dataset

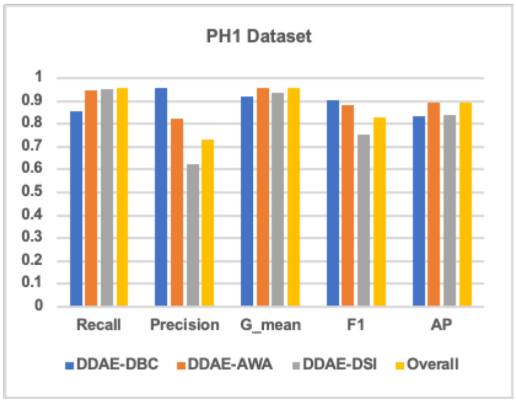


Figure 4.20: Effectiveness of DDAE on PH1 Dataset

4.3.2 Impact of parameters

In this section, the impact of parameters(including the number of blocks in DBC, the relative weight between pull and push in DSI, the unstable ratio and the cost ratio in AWA) on the DDAE model is analyzed through a set of experiments. These experiments are conducted on the Cm1 and Mw1 dataset.

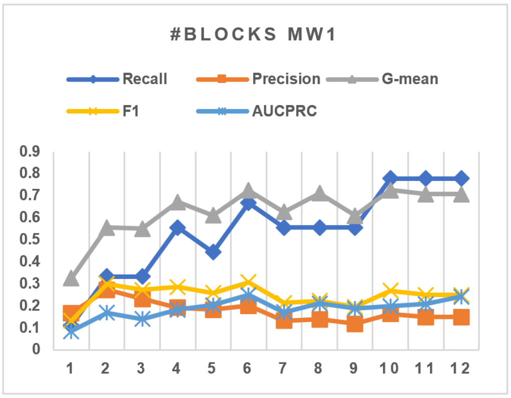


Figure 4.21: Impact of the Number of Data Blocks on CM1

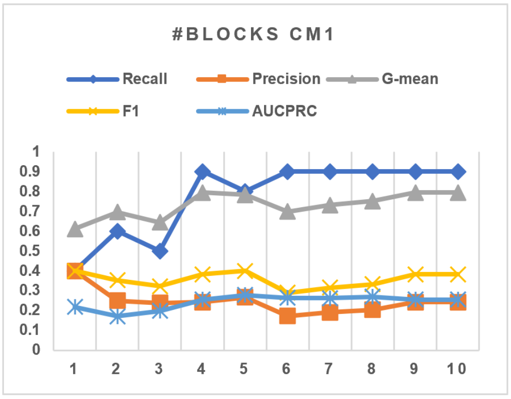


Figure 4.22: Impact of the Number of Data Blocks on MW1

Impact of the number of split data blocks in DBC

The data blocks in DBC are utilized to adjust the imbalanced sample distribution so that each data block can reach a balanced level. As illustrated in Figure 4.21 and Figure 4.22, a similar trend is shared among these evaluation metrics as the number of data blocks grows. It can be observed that the model works better when the number of data blocks is close to the imbalance ratio(IR(Cm1) = 10, IR(Mw1) = 12).

Impact of the relative weight between the pull and push terms in DSI

In this section, the impact of relative weight ω between the pull and push terms in DSI is analyzed by varying ω from 0.1 to 1 on the Cm1 and Mw1 dataset. Notably, the Figure 4.23 and Figure 4.24 illustrate how the model achieves the best performance on Cm1 when $\omega = 0.1$ and on Mw1 when $\omega = 0.3$. Due to the specific data distribution on each dataset, the best choice of ω can differ.

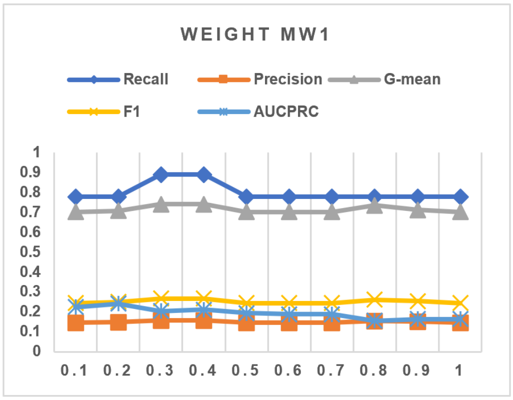


Figure 4.23: Impact of the relative Weight between the Pull and Push Terms on MW1

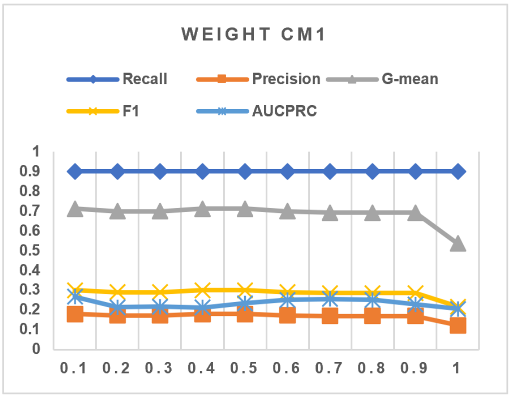


Figure 4.24: Impact of the relative Weight between the Pull and Push Terms on CM1

Impact of the unstable ratio in AWA

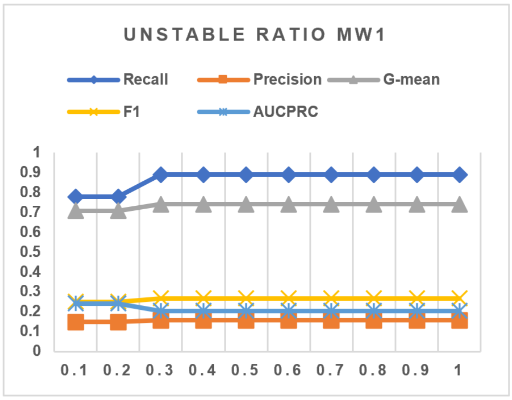


Figure 4.25: Impact of Unstable Ratio on CM1

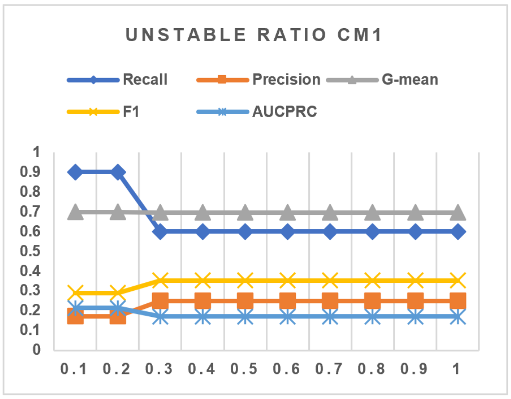
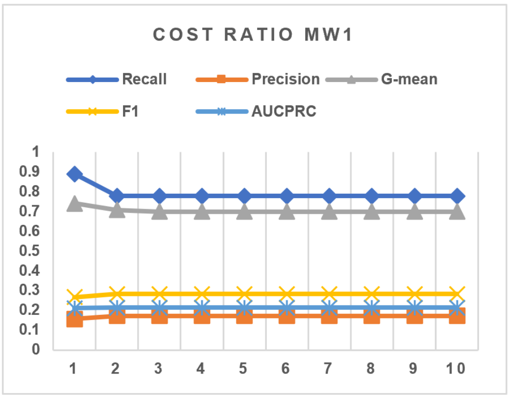


Figure 4.26: Impact of Unstable Ratio on MW1

The unstable ratio τ from AWA is a threshold term that is utilized to help determine whether the AWA generated weight pair should be considered in the decision or not. In order to locate the optimal value for τ, it is varied from 0.1 to 1.0 on the two given datasets. It can be clearly observed from Figure 4.25 and Figure 4.26 that, from $\tau = 0.3$, the trend of these four evaluation metrics becomes stable and unchanged. The model achieves the best performance on the Cm1 and Mw1 datasets when $\tau = 0.2$ and $\tau = 0.3$, respectively.

Impact of the cost ratio in AWA



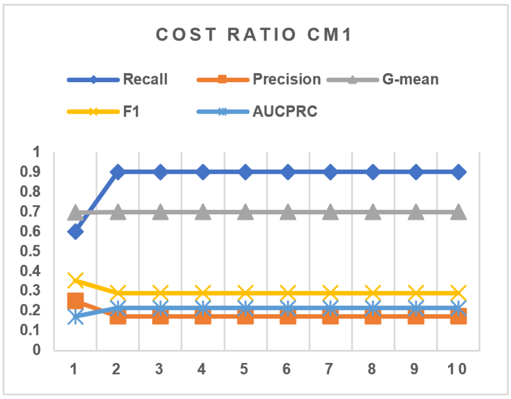


Figure 4.27: Impact of Cost Ratio on MW1

Figure 4.28: Impact of Cost Ratio on CM1

As introduced in Chapter 2, the most significant step of the cost-sensitive learning method is choosing the cost for False Positive(FP) and False Negative(FN), and the cost ratio shows which class is more important in the classification process. But if there is no experts' suggestions or any other additional information related to these two classes, the decision of an appropriate cost ratio may become difficult. In this experiment, the tested range of cost ratio is from 1 to the imbalance ratio. For instance, cost ratio = 1 means the two classes are equally important. From Figure 4.27 and Figure 4.28, it can be seen that, when the cost ratio is above 2, the performance of the model remains stable on both datasets.

Chapter 5

Discussion and Conclusion

The objective of this paper was to compare the performance of various class imbalance classification models when dealing with imbalanced medical/healthcare datasets. For this, DDAE, MWMOTE, SMOTE, RUSBoost, AdaBoost, cost-sensitive decision tree (csDCT), MetaCost, CAdaMEC, self-paced Ensemble Classifier and Iterative Metric Learning (IML) were utilized to classify some testing samples from various imbalanced medical/healthcare datasets. All the imbalanced datasets came from UCI, KEEL and OpenML with different imbalance ratios (IR). The experiments were divided into two stages. In the first stage, each model was used to deal with each given dataset. In addition, a further two experiments focused on the impact of various imbalance ratios and the size of datasets on the performance of models. The second stage focused on the impact of parameters on the DDAE model and the effectiveness of its different components.

The results of the experiments carried out for this paper are reported in Chapter 4. This section contains the overall discussion of the results and influencing factors of these models.

DDAE yielded the highest value for recall on almost all imbalanced datasets, not only on medical/healthcare data but also on datasets from other sectors. Since the recall only focuses on minority, a higher recall means that most samples with a class label of 1 can be correctly predicted. In the medical or healthcare sector, successfully detecting as many people in genuine need of medical intervention can be the most critical goal. However, this does not mean that recall is the most significant evaluation metric when the classifier is used on medical datasets. For example, if DDAE is used to test whether a person has infected by Corona Virus, even though all the infected cases can be screened out (meaning recall is 1), a low precision means that, compared with actual patients, several times more healthy people will be incorrectly diagnosed as infected. This is undesirable at this stage, and may even cause social catastrophe. According to the description of PRC, it can be seen that the recall and precision cannot be satisfactory at the same time, so it is critical for a class imbalance classification model to achieve an appropriate F1. Comparing with other algorithms like csDCT, RUSBoost and self-paced Ensemble Classifier, the F1 of DDAE is extremely low on some datasets. This illustrates that DDAE is more sensitive than others, so it is easier to predict a sample with a class label of 0 as 1, which can be considered first because the number of training samples is too small. On several datasets, such as Poker8vs6, DDAE's F1 performs extremely poorly compared to other algorithms but has better recall performance. This dataset contains a total of 1,477 samples, but its IR is as high as 85.882, that is, only 17 positive samples are included in this dataset. According to the description of DDAE in Chapter 3, the whole training set was divided into 79 data blocks to train each base classifier with each block only consisting of 26 samples (13 positive samples and 13 negative samples). Although each data block reaches a balanced state, when the total number of samples is small and the degree of imbalance is high, the sub-training set obtained by resampling is incomplete. This may cause the model to overfit to minority, fail to learn more robust and easy generalization features, and often have worse generalization performance on highly skewed data. Under these circumstances, if AWA in DDAE assign a non-default weight to minority, the model will become more sensitive to the minority class, which will cause a low precision. Comparing to Poker8vs6, the performance of F1 on PH1 (with 11,274 data samples) and optdigits (with 5,620 data samples) is much better. This can also be confirmed from Figure 4.7 and Figure 4.8, which shows that when the size of dataset is constant, as the IR increases, the F1 and AUCPRC of DDAE on individual dataset shows a significant decrease.

Iterative Metric Learning (IML) is a model that also utilizes Large Margin Nearest Neighbor (LMNN) to improve the data space and uses the kNN classifier as the basic classifier. Unlike DDAE, IML always performs well in F1 and AUCPRC, even though the value of recall is low. It can be observed that the evaluation metrics of IML on different datasets varies, meaning that compared with other models, IML is more sensitive to the data distribution of the dataset. From the results in section 4.2.2, it can be observed that, even the IR climbs up, while the evaluation metrics of IML keep stable.

The performance of SMOTE and MWMOTE are similar to some extent, but it is evident that the value of G-mean, F1 and AUCPRC for SMOTE on some slightly imbalanced datasets, such as WDBC (IR=1.866) and PID (IR=1.684), is better than that of MWMOTE. This can also be observed from Figure 4.7, which shows that the IR is relatively small, SMOTE can achieve a better performance than MWMOTE. SMOTE will randomly select minority samples to synthesize new samples, regardless of the surrounding samples. This may cause the generation of useless samples if the selected minority sample is far away from the decision boundary or the newly generated samples may be overlapped with the majority samples in the surrounding if the selected minority reside inside the majority area [55]. MWMOTE also applies the synthetic sample generation technique. However, unlike SMOTE, MWMOTE utilizes a clustering procedure to ensure that all the produced samples must be located within the minority region to avoid any false or noisy synthetic samples [24]. MWMOTE also finds a more practical approach to select samples that are difficult to learn, and then an appropriate weight will be assigned to them [24], which SMOTE does not consider. This can explain why MWMOTE performs better than SMOTE under circumstances where the IR is high.

RUSBoost, AdaBoost and self-paced Ensemble Classifier are three alternative ensemble learning techniques to DDAE. It should be noted that the performance of RUSBoost and AdaBoost is very similar. Compared with RUSBoost, the self-paced Ensemble Classifier and AdaBoost show a more stable performance. Even though RUSBoost and AdaBoost achieve 0.214 and 0.196 respectively in terms of recall on the Pc3 dataset, the recall of the self-paced Ensemble Classifier is 0.786 for this dataset, which yielding the highest value among all models compared. AdaBoost is a basic implementation of the ensemble learning technique. RUSBoost adds a random undersampling technique on the foundation of boosting. Like DDAE, the self-paced Ensemble Classifier also cuts the majority of the dataset into several bins and uses the resampled balance training set (including majority subset and minority subset) to train each base classifier. The self-paced Ensemble Classifier, RUSBoost and AdaBoost, when compared with DDAE, do not take the weight of both classes into concern, which can also contribute to a lower recall value but higher F1 and AUCPRC values.

Moreover, the remaining of the algorithms are all implementations of cost-sensitive learning, namely MetaCost, csDCT and CAdaMEC. It can be observed that these models can perform well when the IR of the dataset is relatively low, such as in the Euthyroid Sick, optdigits and PH1 datasets. The performance of MetaCost varies when it deals with different datasets. It is hard to set an appropriate cost matrix for each dataset, as people cannot be experts all the time. If the set cost matrix is not suitable for the predicted dataset and the data is highly skewed, the performance of the classifier can be extremely unsatisfactory, such as on the Poker8vs6 dataset. Considering this, it is better to utilize this kind of classifier when the imbalance ratio is not very high. This was also confirmed through the experiments in section 4.2.2, which showed that as the imbalance ratio increases, the curves of all evaluation metrics of MetaCost and CAdaMEC fluctuate, but still show a downward trend.

The impact of the size of the dataset on the performance of the model is noticeable. Except for Metacost, which dispalys an irregular fluctuation, the remaining algorithms either continue to perform very well or all the evaluation metrics of the algorithm rise gradually as the number of samples in the dataset increases. For most algorithms, the more samples in the dataset, the more training samples are used to train the model, which can provide a stronger training basis for the model and enable more accurate prediction. Enlarging the scale of the dataset can enhance the completeness of the data, and can also alleviate the over-fitting phenomenon caused by the resampling method, such as in MWMOTE and SMOTE.

From section 4.3.1, it can be seen that the recall and G-mean drop significantly when the DBC component has been removed. The variant DDAE-DBC displays the poorest performance among all other variants and the original model. This phenomenon shows that DBC plays a vital role in the process of identifying minority samples. DBC turns the data distribution of each data block into a nearly balanced state, which leads to the improvement of the minority's 'status' so that the model can classify minority samples correctly. However, this component has also cause the problem of overfitting, since the removal of DBC lead to a significant decrease on recall but increase on precision. DBC is also critical because it provides support for AWA and EL components. Due to the different data distribution, AWA components may be 'abandoned' sometimes(which means the default weight pair is assigned to the model), but it does not mean AWA is dispensable. On the PH1 dataset, the recall and G-mean of DDAE-AWA perform less satisfactory than the original model. Considering all these evaluation metrics, the original model should be used under some specific circumstances, for instance, when the misclassifcation of majority samples can be ignored.

Given the importance of the DBC component, the selection of an appropriate number of data blocks is also a critical problem. According to the previous description of DBC in Chapter 3, when the original dataset is divided into a small number of blocks, the data distribution in every single block can still maintain imbalance. Thus, it can be seen that DDAE achieves increased performance when the number of data blocks is close to the imbalance ratio value. Nevertheless, the imbalance ratio is not the only option. Notably, the model can also reach a promising performance when the number of data blocks is 4 and 9 on the Cm1 and 10 and 11 on the Mw1 dataset. This phenomenon illustrates that the original dataset might be divided into a smaller number of blocks instead of the value of IR blocks. From this experiment, it can be seen that the number of base estimators set by ensemble learning algorithms also has an impact on the algorithm performance.

When should the two different classes be pushed away from each other, and when should the same class be pulled closer to each other? The answer to this question may lead to the decision of the relative weight ω between pull and push term in the DSI component. For instance, when the data instances from these two classes overlap to a high degree, it is best to push them away from each other through setting a considerable value of ω. On the contrary, if an area is occupied by the majority class instances, it may be critical to set a smaller value at ω in order to bring the minority instances with the same label closer together. The adoption of the AWA component is decided by a threshold τ called unstable ratio. According to the result in this section, the best τ should be set to 0.2.

In AWA, the cost ratio represents the importance of the costs between the majority and minority classes. To be specific, when the cost ratio is set to 1, the majority class and minority class have an equal 'status', as if both of them, are the king of their kingdom. However, from the results presented in this section, all the metrics become stable when the cost ratio is greater than two on both of the datasets in the experiment, showing that the different selection of cost ratio does not affect the performance of the model. This shows the data sources do not restrict the DDAE. The overall results in section 4.1.1 also confirm that even when the datasets are from different fields, the DDAE can deal with them well. This matches the finding stated in study [29].

In this paper, all the experiments focused on the binary classification. However, some of the algorithms used for comparison are also suitable for multi-class classification, such as AdaBoost,

Technique	Pros	Cons	Example
Oversampling	Balance the class distribution of original dataset	Depend on the distribution of minority samples; Require a lot of additional computing resources; May cause the problem of overfitting	SMOTE MWMOTE DDAE
Undersampling	Balance the class distribution of original dataset	May cause the information loss and the problem of underfitting	DDAE RUSBoost
Cost-sensitive Learning	Take the importance of different class into account	Hard to set an appropriate cost matrix for each dataset	MetaCost CAdaMEC csDCT DDAE
Ensemble Learning	Stable; Make more reliable predictions than the single classifier; Reduce the dispersion of model performance	Absorb not only the advantages but also the disadvantages of used techniques when the final classifier is combined with several techniques	RUSBoost AdaBoost self-paced Ensemble Classifier DDAE
Distance Metric Learning	Result in a stable neighborhood for each sample	Depend on the data distribution in the feature space, when the dataset is highly imbalance with a relatively small size, the performance can be unsatisfactory	IML

Table 5.1: Pros and Cons of different Techniques utilized for dealing with Class Imbalance Problem

MetaCost, cost-sensitive decision tree. SMOTE, MWMOTE and IML focuses on the feature space. Thus, if these techniques are utilized for multi-class classification, the classifier combined with these techniques should be suitable for multi-class classification.

In summary, this paper investigated the performance comparison of several class imbalance classification models using medical/healthcare datasets and also some datasets from other fields. Table 5.1 shows the characteristics of techniques used for dealing with class imbalance problem. Since algorithms have various characteristics, it is important to choose the appropriate algorithm depending on the actual situation. For instance, for the algorithms used for stocks prediction, people should pay more attention to their predictions. However, in medical diagnosis or earthquake prediction sector, the recall for the algorithms is more significant. The performance of algorithms cannot be judged by the individual evaluation metric, the actual request and situation should also be taken into account. In addition, the number of base estimators set by ensemble learning classifiers can also make an impact on their performance. This has already been researched with regard to DDAE, but more detail surrounding the impact of the number of base estimators on other ensemble classifiers remains an area for the future research.

Bibliography

[1] J. F. Kong, "What is a class imbalance classification problem?" https://ecole-itn.eu/ecole-blog/#1574339393650-53e7d5da-805b, 2009.

[2] J. Ahn, M. Park, H.-S. Lee, S. J. Ahn, S.-H. Ji, K. Song, and B.-S. Son, "Covariance effect analysis of similarity measurement methods for early construction cost estimation using case-based reasoning," *Automation in Construction*, vol. 81, pp. 254–266, 2017.

[3] Z. Zheng, Y. Cai, and Y. Li, "Oversampling method for imbalanced classification," *Computing and Informatics*, vol. 34, no. 5, pp. 1017–1037, 2016.

[4] C. L. Castro and A. P. Braga, "Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data," *IEEE transactions on neural networks and learning systems*, vol. 24, no. 6, pp. 888–899, 2013.

[5] S. L. Salzberg, "C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993," 1994.

[6] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[7] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification. ieee trans inf theory it-13(1):21-27," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[8] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[9] X. Guo, Y. Yin, C. Dong, G. Yang, and G. Zhou, "On the class imbalance problem," in *2008 Fourth International Conference on Natural Computation*, vol. 4, 2008, pp. 192–201.

[10] 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.

[11] S. Huda, J. Yearwood, H. F. Jelinek, M. M. Hassan, G. Fortino, and M. Buckland, "A hybrid feature selection with ensemble classification for imbalanced healthcare data: A case study for brain tumor diagnosis," *IEEE access*, vol. 4, pp. 9145–9154, 2016.

[12] L. Gao, L. Zhang, C. Liu, and S. Wu, "Handling imbalanced medical image data: A deep-learning-based one-class classification approach," *Artificial Intelligence in Medicine*, vol. 108, p. 101935, 2020.

[13] C. Elkan, "The foundations of cost-sensitive learning," in *International joint conference on artificial intelligence*, vol. 17, no. 1. Lawrence Erlbaum Associates Ltd, 2001, pp. 973–978.

[14] C. Sammut and G. I. Webb, *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[15] P. Branco, L. Torgo, and R. Ribeiro, "A survey of predictive modelling under imbalanced distributions," *arXiv preprint arXiv:1505.01658*, 2015.

[16] 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)*. IEEE, 2010, pp. 1–8.

[17] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[18] L. Rokach, *Pattern classification using ensemble methods*. World Scientific, 2010, vol. 75.

[19] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[20] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "Adacost: misclassification cost-sensitive boosting," in *Icml*, vol. 99. Citeseer, 1999, pp. 97–105.

[21] C. Seiffert, T. M. Khoshgoftaar, J. Van 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, 2009.

[22] M. Zhu, J. Xia, X. Jin, M. Yan, G. Cai, J. Yan, and G. Ning, "Class weights random forest algorithm for processing class imbalanced medical data," *IEEE Access*, vol. 6, pp. 4641–4652, 2018.

[23] N. Liu, X. Li, E. Qi, M. Xu, L. Li, and B. Gao, "A novel ensemble learning paradigm for medical diagnosis with imbalanced data," *IEEE Access*, vol. 8, pp. 171 263–171 280, 2020.

[24] S. Barua, M. M. Islam, X. Yao, and K. Murase, "Mwmote–majority weighted minority over-sampling technique for imbalanced data set learning," *IEEE Transactions on knowledge and data engineering*, vol. 26, no. 2, pp. 405–425, 2012.

[25] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999, pp. 155–164.

[26] N. Nikolaou, N. Edakunni, M. Kull, P. Flach, and G. Brown, "Cost-sensitive boosting algorithms: Do we really need them?" *Machine Learning*, vol. 104, no. 2, pp. 359–384, 2016.

[27] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from imbalanced data sets*. Springer, 2018, vol. 11.

[28] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[29] J. Yin, C. Gan, K. Zhao, X. Lin, Z. Quan, and Z.-J. Wang, "A novel model for imbalanced data classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6680–6687.

[30] N. Wang, X. Zhao, Y. Jiang, Y. Gao, and K. BNRist, "Iterative metric learning for imbalance data classification." in *IJCAI*, 2018, pp. 2805–2811.

[31] Z. Liu, W. Cao, Z. Gao, J. Bian, H. Chen, Y. Chang, and T.-Y. Liu, "Self-paced ensemble for highly imbalanced massive data classification," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 841–852.

[32] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

[33] A. Orriols-Puig and E. Bernadó-Mansilla, "Evolutionary rule-based systems for imbalanced data sets," *Soft Computing*, vol. 13, no. 3, pp. 213–225, 2009.

[34] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi, "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance," *Neural networks*, vol. 21, no. 2-3, pp. 427–436, 2008.

[35] M. Zeng, B. Zou, F. Wei, X. Liu, and L. Wang, "Effective prediction of three common diseases by combining smote with tomek links technique for imbalanced medical data," in *2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS)*. IEEE, 2016, pp. 225–228.

[36] 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, 2008.

[37] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*. Springer, 2005, pp. 878–887.

[38] I. Tomek, "Two modifications of cnn," *IEEE Transactions on Systems Man & Cybernetics*, vol. SMC-6, no. 11, pp. 769–772, 1976.

[39] C. X. Ling and V. S. Sheng, "Cost-sensitive learning and the class imbalance problem," *Encyclopedia of machine learning*, vol. 2011, pp. 231–235, 2008.

[40] Y. Sahin, S. Bulkan, and E. Duman, "A cost-sensitive decision tree approach for fraud detection," *Expert Systems with Applications*, vol. 40, no. 15, pp. 5916–5923, 2013.

[41] K. M. Ting, "Inducing cost-sensitive trees via instance weighting," in *European symposium on principles of data mining and knowledge discovery*. Springer, 1998, pp. 139–147.

[42] J. Surowiecki and M. P. Silverman, "The wisdom of crowds," *American Journal of Physics*, vol. 75, no. 2, pp. 190–192, 2005.

[43] B. V. Dasarathy and B. V. Sheela, "A composite classifier system design: Concepts and methodology," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 708–713, 1979.

[44] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[45] L. Han, S. Luo, J. Yu, L. Pan, and S. Chen, "Rule extraction from support vector machines using ensemble learning approach: an application for diagnosis of diabetes," *IEEE journal of biomedical and health informatics*, vol. 19, no. 2, pp. 728–734, 2014.

[46] C. Merkwirth, H. Mauser, T. Schulz-Gasch, O. Roche, M. Stahl, and T. Lengauer, "Ensemble methods for classification in cheminformatics," *Journal of chemical information and computer sciences*, vol. 44, no. 6, pp. 1971–1978, 2004.

[47] P. Yang, Y. Hwa Yang, B. B Zhou, and A. Y Zomaya, "A review of ensemble methods in bioinformatics," *Current Bioinformatics*, vol. 5, no. 4, pp. 296–308, 2010.

[48] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[49] B. Efron, *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982.

[50] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and systems magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[51] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[52] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[53] N. Nikolaou and G. Brown, "Calibrating adaboost for asymmetric learning," in *International Workshop on Multiple Classifier Systems*. Springer, 2015, pp. 112–124.

[54] H. He and Y. Ma, "Imbalanced learning: foundations, algorithms, and applications," 2013.

[55] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[56] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *European conference on principles of data mining and knowledge discovery*. Springer, 2003, pp. 107–119.

[57] K. M. Ting, "A comparative study of cost-sensitive boosting algorithms," in *In Proceedings of the 17th International Conference on Machine Learning*. Citeseer, 2000.

[58] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *International journal of pattern recognition and artificial intelligence*, vol. 23, no. 04, pp. 687–719, 2009.

[59] A. Luque, A. Carrasco, A. Martín, and A. de las Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix," *Pattern Recognition*, vol. 91, pp. 216–231, 2019.

[60] C. Ferri, J. Hernández-Orallo, and R. Modroiu, "An experimental comparison of performance measures for classification," *Pattern Recognition Letters*, vol. 30, no. 1, pp. 27–38, 2009.

[61] M. Hossin and M. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, p. 1, 2015.

[62] C.-T. Su and Y.-H. Hsiao, "An evaluation of the robustness of mts for imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 19, no. 10, pp. 1321–1332, 2007.

[63] Y. Peng, C. Li, K. Wang, Z. Gao, and R. Yu, "Examining imbalanced classification algorithms in predicting real-time traffic crash risk," *Accident Analysis & Prevention*, vol. 144, p. 105610, 2020.

[64] M. Kubat, R. Holte, and S. Matwin, "Learning when negative examples abound," in *European Conference on Machine Learning*. Springer, 1997, pp. 146–153.

[65] G. H. Nguyen, A. Bouzerdoum, and S. L. Phung, "Learning pattern classification tasks with imbalanced data sets," *Pattern recognition*, pp. 193–208, 2009.

[66] D. Ballabio, F. Grisoni, and R. Todeschini, "Multivariate comparison of classification performance measures," *Chemometrics and Intelligent Laboratory Systems*, vol. 174, pp. 33–44, 2018.

[67] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[68] ——, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[69] J. Davis and M. Goadrich, “The relationship between precision-recall and roc curves,” in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 233–240.

[70] M. Richardson and P. Domingos, “Markov logic networks,” *Machine learning*, vol. 62, no. 1-2, pp. 107–136, 2006.

[71] I. Waspada, N. Bahtiar, P. W. Wirawan, and B. D. A. Awan, “Performance analysis of isolation forest algorithm in fraud detection of credit card transactions,” *Khazanah Informatika: Jurnal Ilmu Komputer dan Informatika*, vol. 6, no. 2, 2020.

[72] I. Plyusnin, L. Holm, and P. Törönen, “Selection of evaluation metrics for gene ontology classifiers: Supplementary information.”

[73] K. Q. Weinberger and L. K. Saul, “Distance metric learning for large margin nearest neighbor classification.” *Journal of machine learning research*, vol. 10, no. 2, 2009.

[74] S. Shalev-Shwartz, Y. Singer, and A. Y. Ng, “Online and batch learning of pseudo-metrics,” in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 94.

[75] J. Goldberger, G. E. Hinton, S. Roweis, and R. R. Salakhutdinov, “Neighbourhood components analysis,” *Advances in neural information processing systems*, vol. 17, pp. 513–520, 2004.

[76] P. C. Mahalanobis, “On the generalized distance in statistics.” National Institute of Science of India, 1936.

[77] C. Zhang and Y. Ma, *Ensemble machine learning: methods and applications*. Springer, 2012.

[78] D. Dua and C. Graff, “Uci machine learning repository,” http://archive.ics.uci.edu/ml, 2017.

[79] K. C. 2004, “Particle physics; plus protein homology prediction,” https://www.kdd.org/kdd-cup/view/kdd-cup-2004/Data, 2004.

[80] T. Menzies, J. Greenwald, and A. Frank, “Data mining static code attributes to learn defect predictors,” *IEEE transactions on software engineering*, vol. 33, no. 1, pp. 2–13, 2006.

[81] J. Alcalá-Fdez, A. Fernández, 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 & Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2011.

[82] S. Krishnan, J. Wang, M. J. Franklin, K. Goldberg, T. Kraska, T. Milo, and E. Wu, “Sampleclean: Fast and reliable analytics on dirty data.” *IEEE Data Eng. Bull.*, vol. 38, no. 3, pp. 59–75, 2015.

[83] F. Kamiran and T. Calders, “Data preprocessing techniques for classification without discrimination,” *Knowledge and Information Systems*, vol. 33, no. 1, pp. 1–33, 2012.

[84] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*. Springer International Publishing, 2015.

[85] J. Han, M. Kamber, and J. Pei, “Data mining concepts and techniques third edition,” *The Morgan Kaufmann Series in Data Management Systems*, vol. 5, no. 4, pp. 83–124, 2011.

[86] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas, “Data preprocessing for supervised leaning,” *International Journal of Computer Science*, vol. 1, no. 2, pp. 111–117, 2006.

[87] S. Van Buuren, *Flexible imputation of missing data*. CRC press, 2018.

[88] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O’Reilly Media, 2019.

[89] P. D. Allison, *Missing data*. Sage publications, 2001.

[90] V. Hodge and J. Austin, “A survey of outlier detection methodologies,” *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[91] C. C. Aggarwal, “Outlier analysis,” in *Data mining*. Springer, 2015, pp. 237–263.

[92] H. Yang, “Data preprocessing,” 2018.

[93] M. Kantardzic, *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.

[94] T. Shah, “Validation and test sets in machine learning.” https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7, 2017.