Experimental Comparison of Classification Methods under Class Imbalance

Hui Chen[†]and Mengru Ji[‡]

[†]Beijing Foreign Studies University, China; chenhui@bfsu.edu.cn (corresponding author)
[‡]University of Göttingen, Germany; mengru.ji@stud.uni-goettingen.de

Abstract

The class imbalance problem is prevalent in many domains including medical, natural language processing, image recognition, economic and geographic areas etc. We perform a systematic experimental comparison of different imbalance classification algorithms — ranging from sampling, distance metric learning, cost-sensitive learning to ensemble learning approaches — on several datasets from UCI, KEEL and OpenML. The algorithms included DDAE, MWMOTE, SMOTE, RUSBoost, AdaBoost, cost-sensitive decision tree (csDCT), self-paced Ensemble Classifier, MetaCost, CAdaMEC and Iterative Metric Learning (IML). As the substantial bias potentially caused by imbalance classification can be harmful for underrepresented classes which are of critical social and economic values and even lives, the main objective of our study is thus to understand the impact of imbalance ratio and the size of the utilized datasets on the performance of the above-mentioned algorithms. Our experiments show that 1) Sampling methods perform the worst and cannot be used directly for imbalanced classification, since they lack of consideration of neighborhoods based on distance. However, some classifiers can be improved after the balance of class distribution. 2) Cost-sensitive learning models should be utilized when the dataset is less imbalanced, because it is difficult to set an appropriate cost matrix for a specific dataset, which can cause performance fluctuations. 3) IML consistently shows good performance (in terms of F1 and AUCPRC), is resilient to different imbalance ratios but sensitive to the data distribution of the dataset. 4) Ensemble learning techniques generally perform better over other approaches due to their combined intelligence of multiple basic classifiers. 5) In terms of system performance, self-paced Ensemble Classifier performs fairly well with regards to learning time, while IML and DDAE yield the longest learning time; AdaBoost and self-paced Ensemble Classifier are two algorithms require lowest memory usage. We also provide our empirical recommendation for algorithm selection under different requirements and usage scenarios based on our analysis.

Received on 23 May 2021; accepted on 10 June 2021; published on 11 June 2021

Keywords: Imbalance Classification, Resampling, Cost-Sensitive Learning, Distance Metric Learning, Ensemble Learning, Performance Evaluation

doi:10.4108/eai.11-6-2021.170234

1. Introduction

Classification is one of the most popular topics of machine learning [1–3] and much attention has been paid to binary classification [4–6]. Classical classification methods include Naïve Bayes, k-nearest neighbor (kNN) [7], support vector machine (SVM) [8], decision tree [9] and random forest [10]. These classification algorithms typically assume their datasets are balanced in their class distribution. However, in many real-world application domains such as medical diagnosis [3, 11–14], streaming and social behavior data analysis [2, 15], software development process [16, 17], financial frauds [18–20], unsolicited phone calls [21], disaster risks [22, 23], recommendation systems [24, 25], and text classification [26–28], inherently imbalanced datasets are commonly seen. Also, the limitation of the data collection process and the imbalanced cost for fixing different errors can lead to the imbalance. Affected by these conditions, normal classifiers are often confused by the majority class and ignore the minority class, which may result in catastrophe for instance massive waste of resources, time or money, and even can endanger the lives [29–31]. Additionally, for the percentage of examples available for each class, most real-world datasets 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 [32]. This issue is especially critical since it leads to a significant barrier in the performance achieved by classical learning methods which typically assume that the class distribution is balanced [31, 32]. To combat the losses in critical social and economic values and even lives, its highly important to understand the impact of class imbalance in different imbalance learning algorithms and recommend an appropriate algorithm for a given use scenario.

In practical situations, the ratio between two classes of a dataset may be substantial, similar to 1:100, 1:1000, 1:10000, or even higher [33]. Fig. 1 and Fig. 2 show two dataset examples with different Imbalance Ratios (IR) of 1.684 and 12, respectively.

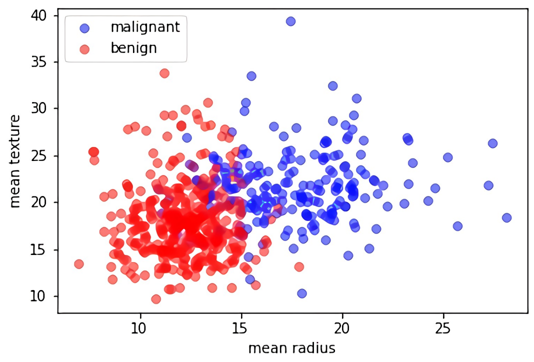

Figure 1. Class distribution on the dataset with IR=1.684

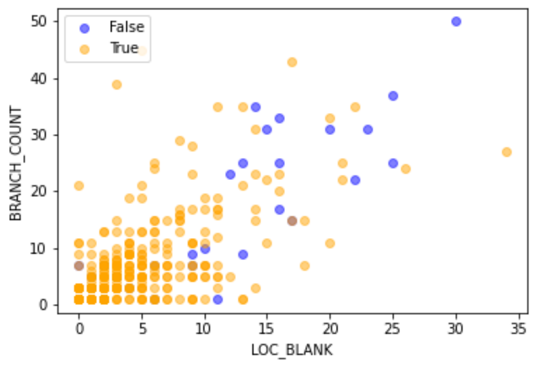

Figure 2. Class distribution on the dataset with IR=12

Solutions to tackle the imbalance problem of classification can be broadly classified into four major families [9]: sampling methods (including oversampling and undersampling) [34, 35], cost-sensitive learning [29], distance metric learning [36], and ensemble learning [37] and hybrid methods which integrate the features from different families such as Adacost [38], RUSBoost [39] and DDAE [40].

As demonstrated in previous literature, a poor-performing algorithm for imbalance classification can lead to substantial losses in critical social and economic values and even lives [12, 14, 19, 20, 23, 27, 28, 31]. Hence, it is important to study the impact of different metrics on the different algorithms under different situations, so that an appropriate choice could be made when deciding the algorithms used for imbalance classification. Although there are evaluation studies on one of the specific directions for imbalance classification or focusing on one specific metric [41], to our best knowledge no work has been conducted using a comprehensive set of evaluation metrics to quantify the performance of representative algorithms from these different classification families.

This paper focuses on comparing the performance of these algorithms on multiple datasets from several different domains, including healthcare, card playing, software development projects and hand-written digit recognition. We use these datasets to evaluate ten imbalanced classification algorithms, namely 1) sampling: SMOTE [35] and MWMOTE [42]; 2) cost-sensitive learning: MetaCost [43], CAdaMEC [44] and cost-sensitive decision tree [9]; 3) distance metric learning: Iterative Metric Learning (IML) [36]; 4) ensemble learning and hybrid methods: AdaBoost [45], RUSBoost [39], self-paced Ensemble Classifier [11] and DDAE [40]. Our experiments not only analyze the performance of different models based on a general set of evaluation metrics on the same dataset, but also quantify the impact of key factors related to imbalanced learning, such as the size of the dataset and the imbalance ratio, as well as system performance in terms of learning time and memory usage.

The following sections will first review the related work, and then present our data sources. In Section 4 are the detailed evaluation results, with additional discussions following in Section 5. Section 6 concludes this paper.

2. Related Work

Due to the vital importance of data analysis for human health, lives and the socioeconomic world, many researchers examined the issue of imbalanced classification. For example, [46] combined feature selection and ensemble classification to solve the problems of imbalanced healthcare data on the diagnosis of a brain tumor. [47] introduced a novel voting class weight algorithm based on random forest algorithm to identify the minority class sufficiently in medical applications and can be applied to the detection of diseases. [48] presented an innovative comprehensive ensemble learning paradigm that involves multiple SVM diversity structures for classification for the early detection of diseases with imbalanced data.

2.1. Sampling Methods

Resampling means creating a new transformed version of the training set of an imbalanced dataset, offering a set of practical and straightforward approaches to provide a more balanced data distribution [34]. The three main resampling methods are oversampling, undersampling, and hybrid techniques which are a combination of both from sampling algorithms [9].

Among these three groups, informed undersampling such as EasyEnsemble and BalanceCasad [49], and synthetic sampling such as SMOTE [35] and the Borderline-SMOTE [50] are shown to outperform random oversampling and random undersampling. However, undersampling methods can lead to information loss which in turn results in loss in classification performance and underfitting, while oversampling suffers from issues of overfitting, high computational overhead and long training time [4].

Synthetic Minority Oversampling Technique (SMOTE). SMOTE [35] addresses the class imbalance problem by generating synthetic samples in feature space (see Fig. 3 for more details). One minority class example s_1 will be selected randomly and then its k nearest neighbors in the 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 [35]. SMOTE creates the synthetic samples through a convex combination of s_1 and s_2 [51]. As described in [52], 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 is proven to outperform the plain undersampling [35].

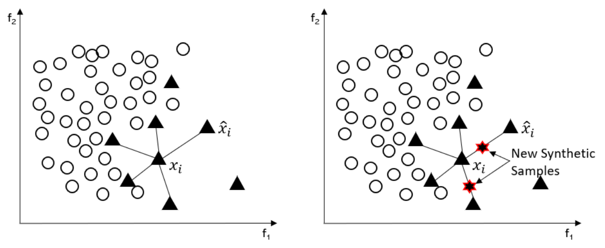

Figure 3. SMOTE working procedure [53]

2.2. Cost-sensitive learning methods

Cost-sensitive learning methods are utilized to deal with different misclassification errors that incur different penalties to find the optimal decision based on the cost matrix [29, 54] as shown in Table 1. If m stands for the predicted label and n stands for the actual label, the $C(m, n)$ is the cost of predicting a class n sample as class m. Given the cost matrix, the purpose of this type of learning method is explained to create a model with minimal overall misclassification costs [55, 56].

Table 1. Cost Matrix for Binary Classification

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

As most traditional classifiers assume that the misclassification has the same cost for false negative (FN) and false positive (FP) [29], the 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 [29, 30, 55]. The supplied cost matrix can strongly impact the effectiveness and a common phenomenon is that, the cost of classification errors cannot be described clearly and domain expert knowledge is lacking [9, 54].

Cost-sensitive learning techniques can be categorized into two families: *meta-learning* (including two aspects: thresholding and sampling) and *Direct methods* [56]. 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 [56]. Although efficient due to their ability to take account of the importance of different classes, a disadvantage of cost-sensitive algorithms is that it is difficult to define an appropriate cost matrix for each dataset and generalize the learning algorithm [57].

Cost-sensitive DeCision Tree (csDCT) [9]. Its main idea is to minimize two separate costs: the test cost of the feature and the cost of misclassification of the sample. It takes account of misclassification during pruning and is popularly used as a direct method for detecting card frauds when the cost to misclassify could vary [56, 58]. *Weighting* [59] is one implementation of the sampling methods, in which examples of the minority are assigned high weights according to their proportion.

CAdaMEC. CAdaMEC [44] is proposed upon AdaMEC [60] (a cost-sensitive algorithm) through an appropriate calibration with Platt scaling.

MetaCost. MetaCost [43] is another cost-sensitive learning model which includes a cost-minimizing method independent of the number of classes or arbitrary cost matrices. It relabels the instances in the training set with the class labels that have estimated minimal costs, then the new replacement training set will be applied to the error-based learners. MetaCost can be viewed as a representative thresholding tool in the case of thresholding (see Section 2.5).

2.3. Distance Metric Learning Methods

Iterative Metric Learning (IML) [36] 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 [36], which locates a stable neighborhood for the specific testing data, using k-nearest neighbors rule (kNN) [7] as the base classifier. IML comprises the following steps: 1) *Large Margin Nearest Neighbor (LMNN)* learning [61] (a learning method which offers a much higher accuracy than kNN; see Fig. 4) to improve the data space (through relabeling and regrouping the neighbouring data blocks) which separates the data samples with a different class label by a large margin and makes the samples with the same class labels close to each other; 2) calculate the distance between each of the samples from the training set to the testing samples; 3) run this and previous steps through multiple iterations, controlled by a predefined matching ratio. [62] shows that IML yields to a classification performance bound but requires the learned matrix conforms to the Positive Semi-Definite (PSD) constraint; its performance is still unknown for non-linear metrics.

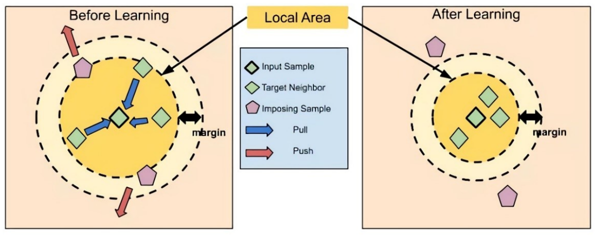

Figure 4. The procedure of LMNN Distance Metric Learning

2.4. Ensemble Learning and Hybrid Methods

In 1979, Dasarathy and Sheela [63] presented one of the earliest studies on ensemble learning, which partitions the feature space with two or more classifiers. In 1990, Hansen and Salmon [12] utilized *ensemble artificial neural networks (ANNs)* with similar configurations to improve a single classifier's generalization performance. In 2005, Surowiecki [64] illustrates the basic idea of various ensemble learning methods and shows that under certain controlled circumstances, the ensemble decisions or predictions of humans often outperform those made by an individual.

The ensemble methodology is used to enhance the individual classifiers. The key idea is to train multiple classifiers and then combine them to achieve an overall classification. The ensemble learning approach has been successfully applied to many areas like medical diagnosis [65, 66], cheminformatics [67, 68], and bioinformatics [69, 70]. Ensemble methods reduce the dispersion of model performance and can make reliable prediction performance, but since they are typically based on either sampling and/or cost-sensitive methods as basic classifiers, they may enjoy the benefits of these basic classifiers but also inherit their disadvantages.

Boosting is one of the most practical techniques of ensemble learning through instance partitioning [71]. Simply put, Boosting generates an ensemble classifier by applying resampling methods on data and is later combined with major voting [64, 72]. In 1997, Freund and Schapire introduced *AdaBoost* (Adaptive Boosting) [45], one of the most representative works of boosting, which applies an iterative process to simple boosting to improve performance. This approach focuses on the instances which are much more complex to classify.

Adaptive Boosting (AdaBoost). *Adaboost* [73] 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. More specifically, in AdaBoost, a new dataset, in which higher weights are assigned to instances that are misclassified by the previous classifier and a lower weight is assigned to the one with a correct prediction, is used for training each subsequent classifier [74]. This is implemented by varying the sample weight, which indicates its importance in the classifier training process, stage by stage. Fig. 5 shows the process of combining the ultimate classifier.

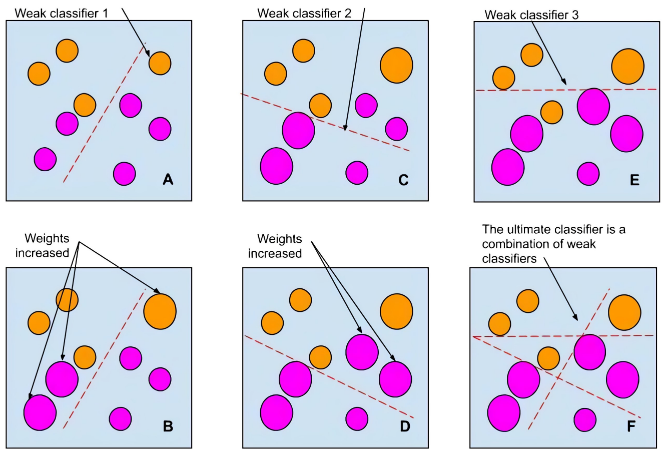

Figure 5. The process of the combination of the ultimate strong learner in *AdaBoost*

Majority Weighted Minority Oversampling Technique (MWMOTE). MWMOTE [42] balances the class distribution also by generating synthetic examples from a weighted minority class through a clustering approach. The weight of each important minority sample is chosen based on its Euclidean distance to the nearest majority class sample.

RUSBoost. RUSBoost [39] is a combination of data sampling and boosting algorithm based on SMOTE-Boost [75] that balances class distribution through SMOTE [35] 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) for performance improvements.

DDAE. DDAE [40] is a novel model to address the class imbalance problem consisting of resampling, data metrics learning, cost-sensitive learning, and ensemble learning. Besides using kNN as the base classifier, DDAE has four components: 1) Data Block Construction (DBC), divides the training (both minority and majority) samples into different number of data blocks based on the given balanced ratio; 2) Data Space Improvement (DSI) applies LMNN (similar to IML) to improve the data space (through relabeling and regrouping the neighbouring data blocks) for training samples in each data block generated in the DBC component; 3) Adaptive Weight Adjustment (AWA) finds an appropriate overall class weight generated using the data coming from each data block [89]; 4) Ensemble Learning (EL) leverages ensemble learning with the weight determined via AWA; multiple base classifiers with major voting technique work on the final decision for each input sample.

Self-paced Ensemble Classifier. Self-paced Ensemble Classifier [11] is an effective ensemble classifier generated by self-paced harmonizing data hardness through undersampling. [11] shows this method can achieve robust performance even when the classes are highly overlapped and the data distribution highly skewed.

2.5. Evaluation metrics

Evaluation of classification performance plays a significant role in guiding and learning performance [76]. Hence, we make extensive evaluation of these existing classification algorithms in different scenarios to understand their pros and cons in this paper.

One popular way of describing evaluation metrics is through the ***confusion matrix*** [77], which provides multiple performance results to offer a deeper understanding of predictive model performance, so that the types of error can be more clearly observed. Table 2 shows the structure of a confusion matrix for binary classification. Table 3 shows the measures derived using the confusion matrix from Table 2.

Table 2. Confusion Matrix for Binary Classification

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

Table 3. Evaluation Metrics based on Confusion Matrix

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*	$\frac{\left(1+\beta^2\right)*\text{ Precision }*\text{ Recall}}{\beta^2*\text{ Precision }+\text{ Recall}}$
Specificity	$\frac{TN}{TN+FP}$
Geometric Mean	$\sqrt{\text{Sensitivity }*\text{ Specificity}}$

A more systematic way of depicting evaluation metrics was proposed by Ferri et al. [78], which classifies the evaluation metrics into three groups: *probability metrics*, *ranking metrics* and *threshold metrics*. In this paper, we apply two groups of metrics, threshold and ranking metrics, 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 [79]. 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 [80].

Sensitivity-specificity metrics are practical thresholding metrics for imbalanced classification that are applied by several researchers [81, 82]. As defined in Table 3, ***specificity*** means the true negative rate. ***Sensitivity***, the complement to specificity, describes the true positive rate. The ***geometric mean (G-mean)*** is calculated through the combination of sensitivity and specificity [83] and 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 [84]. 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 [85]. In this paper, ***F1*** (the value of β in F_β-*Measure* is 1) is utilized as one of the most important evaluation metrics.

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

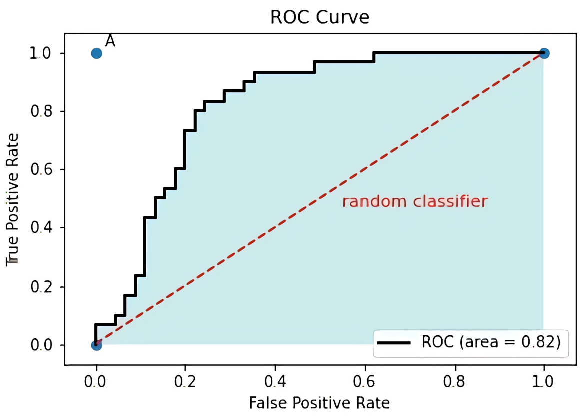

Figure 6. An example of ROC Curve

and the actual data, which can make the performance misleading [51].

The ranking metrics focus on how effectively the base classifiers rank the examples [78]. 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 [9]. ***Receiver Operating Characteristics (ROC) Curve*** [86] 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 [54]. The ***area under the ROC curve (AUCROC)*** is generally applied to measure different classifiers' performance, which is summarized into a single metric [84]. An example of ROC Curve can be observed from Fig 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 [84, 86]. However, in [54], 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 [86]. The ***Precision-Recall (PR) Curve***, which assesses a more informative representation of performance, is utilized to solve such a limitation [54, 87]. PR-Curve is a plot of Recall on the x-axis and Precision on the y-axis [87], 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 [54]. Due to its high level of performance with highly skewed data, it has been applied to the evaluation of performance by many researchers, such as [88–90]. 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 $C(1,1)$) [9]. Similar to AUCROC, the ***area under the PR Curve (AUCPRC)*** is also a summary of PR-Curve with a single scale value [9]. An example of PR-Curve can be observed in Fig 7.

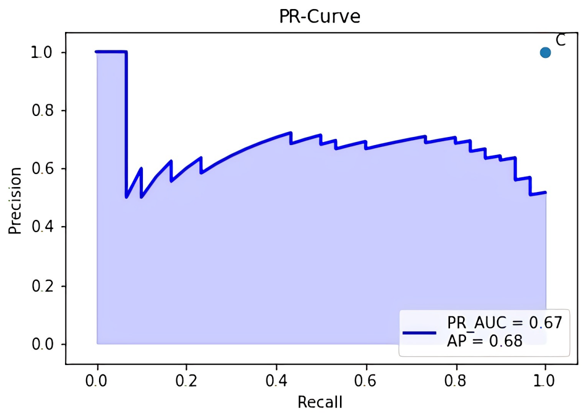

Figure 7. An example of PR Curve

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

3. Data Sources

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 [91], and two sub datasets of Protein Homology (PH1 and PH2) from KDD Cup 2004 [92], which are utilized to test the performance of these models. The detail of these datasets is depicted in Table 4. In addition, eight further datasets are employed, including Cm1, Mw1, Pc1, Pc3, Pc4 which are from NASA Metrics Data Program (NASA) dataset [93] on the software development process, two datasets Poker89vs6 and Poker8vs6 for card playing, which are from KEEL [16], 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 5.

We apply a few data cleaning techniques introduced by previous works (e.g., [94–97]). Data entries with duplicated, inconsistent, or missing values are either deleted or corrected by replacing the missing value with

Table 4. Characteristics of Used Healthcare Datasets

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	14.300
WDBC	2	768	8	1.866
PID	2	568	32	1.684

Table 5. Characteristics of Used Datasets from Other Fields

Dataset	#Class	#Instances	#F	IR
optdigits	2	5,620	65	9.144
Cm1	2	497	21	9.354
Mw1	2	403	37	12.000
Pc1	2	1,109	21	13.400
Pc3	2	1,563	37	8.769
Pc4	2	1,458	37	7.191
Poker89vs6	2	1,485	10	58.400
Poker8vs6	2	1,477	10	85.882

a dummy value or the median of the corresponding attribute, depending on the nature and distribution of the data. For instance, in the Euthyroid Sick Dataset, the missing rate of the attribute 'Age' is 14.10% (466 of 3163). Fig. 8 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 [98] 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[1]', 'F[2]', '?[3]' (see Fig. 9, so this column can be converted into IS_SEX_MALE, IS_SEX_FEMALE, IS_SEX_NA).

Outliers with outstanding values are identified and cleaned up, with the aid of data visualization e.g., by box plots with quantiles. For example, Fig. 10 and Fig. 11 depict the box plots for the attributes 'Blood Pressure' and 'Glucose' from the PID dataset, respectively. 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.

Further used data preprocessing techniques include feature encoding [99], which allows data transformation to make data more acceptable as input for models.

[1]Male
[2]Female
[3]Missing Value

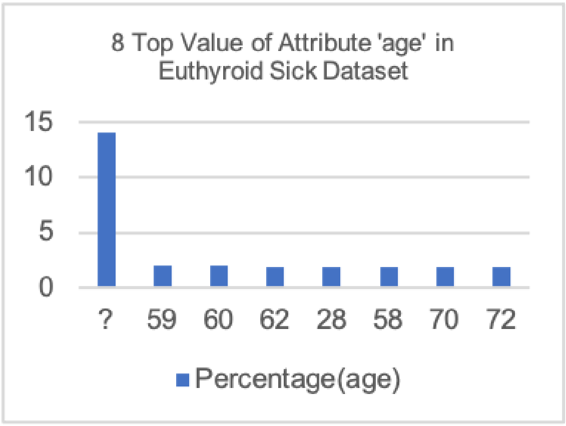

Figure 8. Age Attribute in Euthyroid Sick Dataset

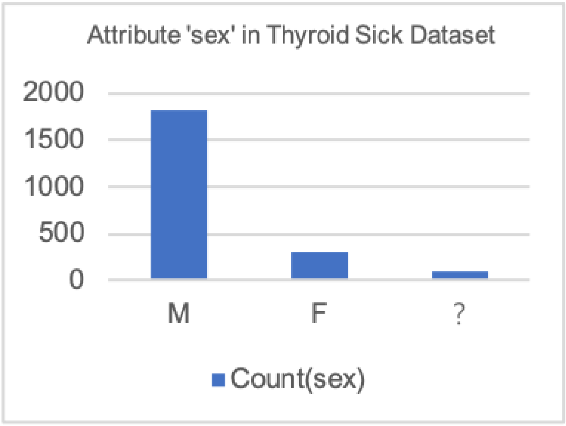

Figure 9. SEX Attribute in Thyroid Sick Dataset

For example, the attribute 'referral source' of thyroid sick dataset contains five values: 'SVI', 'SVHC', 'STMW', 'SVHD' and 'other'. Table 6 shows the results of using feature encoding for this attribute before and after transformation. This attribute will be replaced by four new attributes: 'RSrc1', 'RSrc2', 'RSrc3' and 'RSrc4'. The package Scikit-Learn provides a OneHotEncoder class to transform these kinds of category values into one-hot vectors [99].

Table 6. Comparison of Before and After One Hot Encoding for Attribute 'referral source' in Thyroid Sick Dataset

Raw No	referral source
1	SVI
2	STMW
3	SVHD
4	SVHC
5	other

$\Longrightarrow$

RSrc1	RSrc2	RSrc3	RSrc4
1	0	0	0
0	0	1	0
0	0	0	1
0	1	0	0
0	0	0	0

In our experiments, all datasets are split based on $TrainingSet : TestSet = 7{:}3$ and all experiments are run 10 times and all the performance values are averaged.

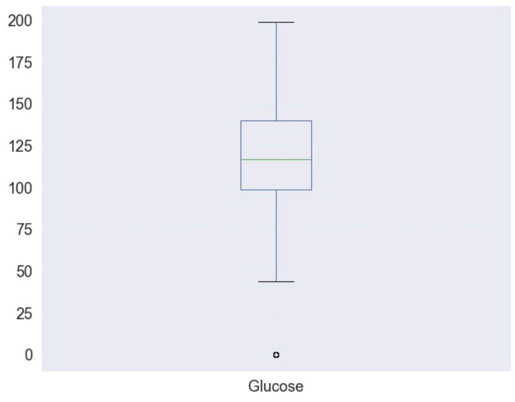

Figure 10. Box Plot for Attribute 'Glucose' in PID Dataset

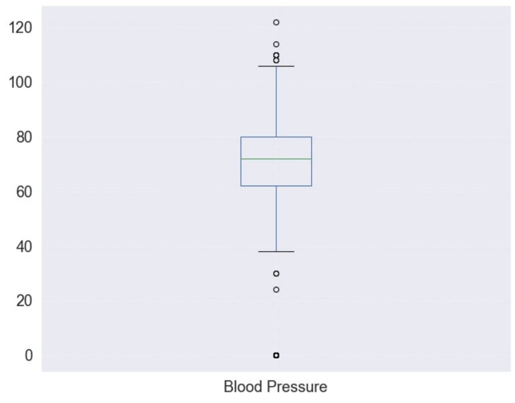

Figure 11. Box Plot for Attribute 'Blood Pressure' in PID Dataset

4. Results

4.1. Overall Comparison

Tables 7-21 present the results of our performance evaluation of all examined imbalance classification algorithms in terms of G-mean, F1 and AUCPRC for each separate dataset.

As shown from these results, 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, DDAE's F1 and AUCPRC are not satisfactory; they are, lower than the corresponding average values on most of the datasets. In contrast, AdaBoost and self-paced Ensemble Classifier yield high performance in terms of F1, AUCPRC and G-mean under most circumstances. These two ensemble algorithms seem can accurately

Table 7. Overall Results for Euthyroid Sick Dataset

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 8. Overall Results for Thyroid 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 9. Overall Results for PH1 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 10. Overall Results for PH2 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	**1.000**
MetaCost	0.884	0.794	0.783
csDCT	**0.999**	**0.999**	0.997
CAdaMEC	0.995	0.991	**1.000**
self-paced	**0.999**	**0.999**	**1**
IML	0.978	0.964	0.999
Average	0.972	0.892	0.878

Table 11. Overall Results for optdigits 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 12. Overall Results for MGC 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 13. Overall Results for WDBC 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 14. Overall Results for PID 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 15. Overall Results for Yeast1vs7 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	0.000	0.000	0.129
CAdaMEC	0.000	0.000	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 16. Overall Results for Poker8vs6 Dataset

Models	Poker8vs6		
	G-mean	**F1**	**AUCPRC**
DDAE	0.622	0.029	0.078
MWMOTE	0.866	0.857	0.752
SMOTE	0.999	0.889	0.800
RUSBoost	0.707	0.667	0.761
AdaBoost	**1.000**	**1.000**	**1**
MetaCost	0.000	0.000	0.000
csDCT	0.443	0.019	0.009
CAdaMEC	0.000	0.000	0.007
self-paced	0.613	0.085	0.038
IML	0.500	0.400	0.379
Average	0.695	0.395	0.482

Table 17. Overall Results for Poker89vs6 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	**1.000**	**1.000**	**1.000**
AdaBoost	**1.000**	**1.000**	**1.000**
MetaCost	0.000	0.000	0.015
csDCT	0.495	0.032	0.019
CAdaMEC	0.000	0.000	0.014
self-paced	0.567	0.036	0.016
IML	0.816	0.800	0.848
Average	0.755	0.382	0.559

Table 18. Overall Results for Cm1 Dataset

Models	Cm1		
	G-mean	**F1**	**AUCPRC**
DDAE	0.699	0.290	0.214
MWMOTE	0.679	0.286	0.182
SMOTE	0.715	0.308	0.219
RUSBoost	0.438	0.250	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 19. Overall Results for Pc1 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 20. Overall Results for Pc4 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 21. Overall Results for Pc3 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

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, RUSBoost shows relatively stable performance and close to the average on most datasets.

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.

To investigate the ability of theses models on positive sample detection, the recall for all these models on the medical datasets is presented in Table 22.

It shows 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.

4.2. Impact of Imbalance Ratio

First, to analyze the influence of the 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 12-14 is IR.

Fig. 12 illustrates the trend of recall of all algorithms as the IR increases. 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 remains nearly stable except MetaCost. All of the models still retain a high value for G-mean even when the IR is 70 or 80. Fig. 13 shows 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.

F1 is an evaluation metric which takes both recall and precision into account. In other words, if the

Table 22. Recall for for All Models on 8 Medical Public Datasets

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	**1.000**	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	0.000	**0.750**	**0.750**	0.375

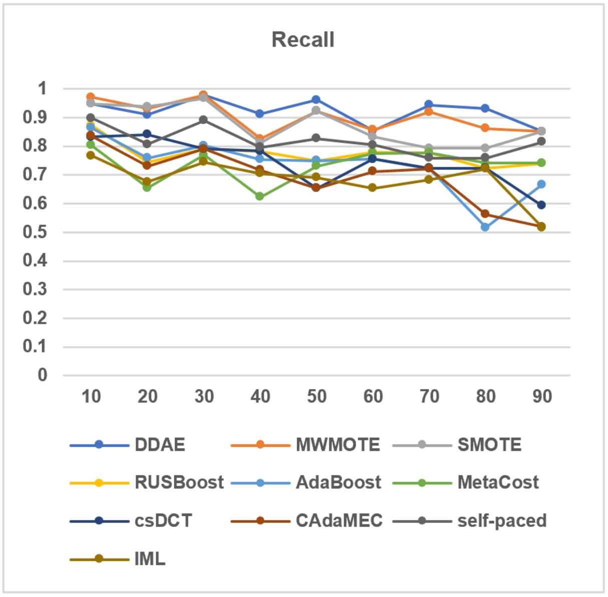

Figure 12. Recall: Impact of Imbalance Ratio

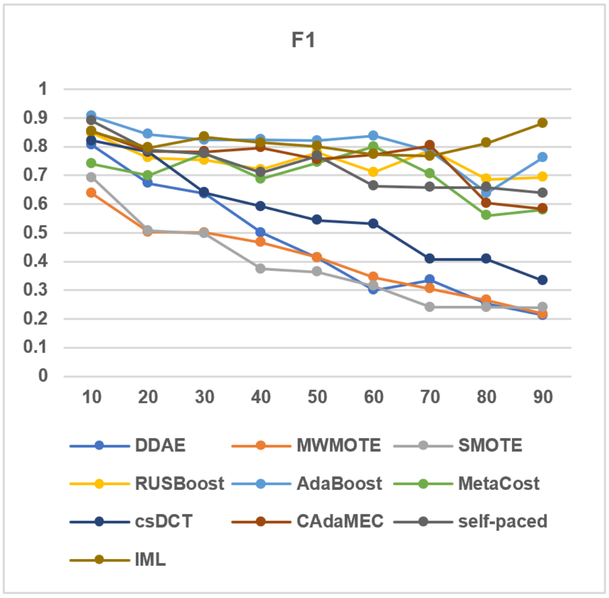

Figure 14. F1: Impact of Imbalance Ratio

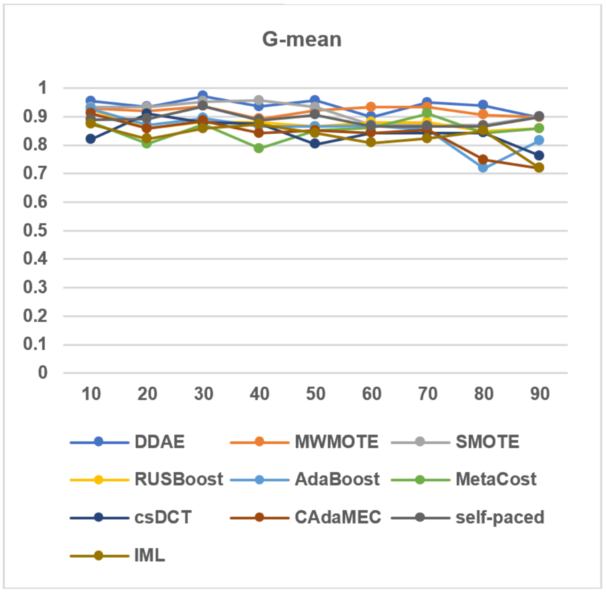

Figure 13. G-mean: Impact of Imbalance Ratio

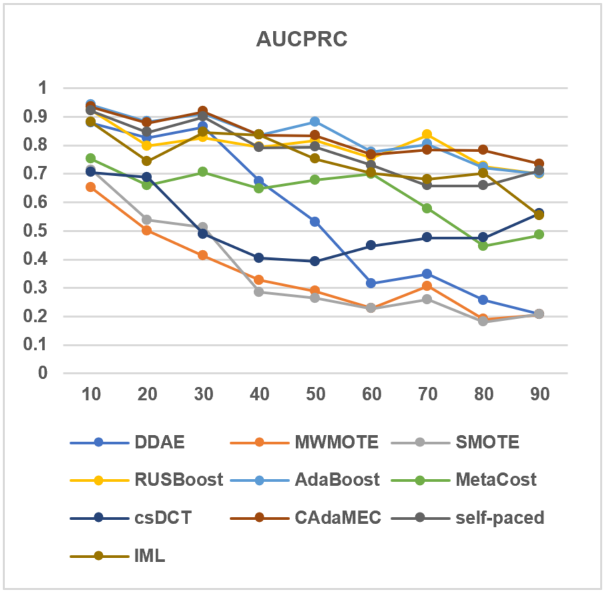

Figure 15. AUCPRC: Impact of Imbalance Ratio

trend of recall for a specific model maintains stability with only a slight decrease/increase, the changes in

F1 will be similar to those in precision. Fig. 14 shows F1's trends of SMOTE, MWMOTE, DDAE and csDCT, which reflects the changes in precision. The value of F1 for MetaCost and AdaBoost decreases 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 the 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. As shown in Fig. 15, 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.

4.3. Impact of the Size of Datasets

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 different numbers 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 16-25.

It can be seen that with the sample dataset sizes, the values of the five evaluation metrics for the algorithms DDAE, MWMOTE, SMOTE, IML and cost-sensitive Decision Tree (csDCT) have an upward trend, especially the precision and F1, which increase 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 is quite excellent, and the values 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 classes correctly among all the algorithms.

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

4.4. Evaluation on System Performance

We also evaluate the system performance of these algorithms. All our experiments are conducted on

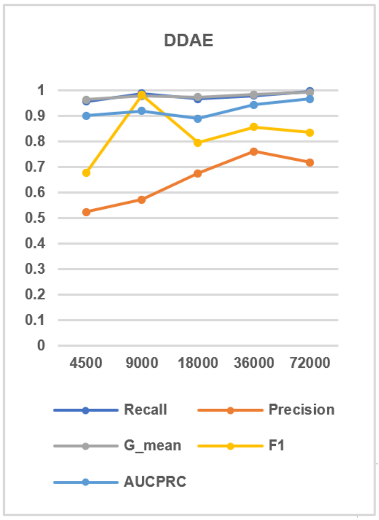

Figure 16. DDAE: Impact of Dataset Size

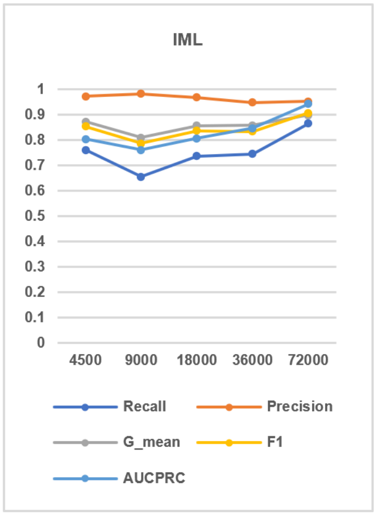

Figure 17. IML: Impact of Dataset Size

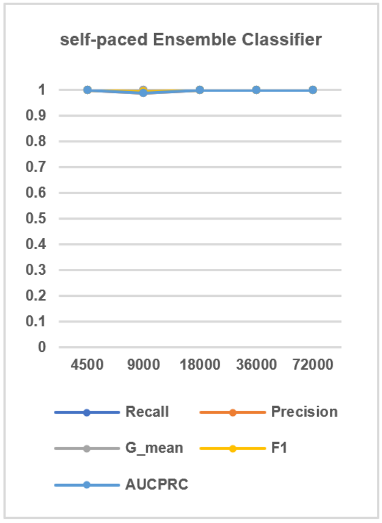

Figure 18. self-paced: Impact of Dataset Size

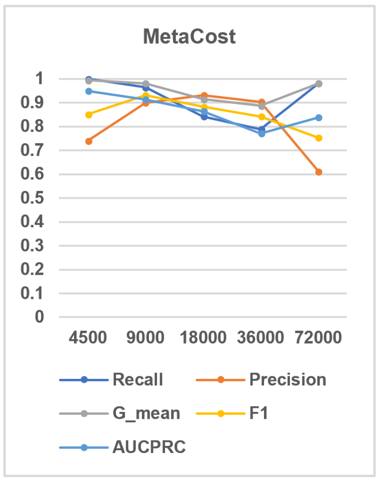

Figure 19. MetaCost: Impact of Dataset Size

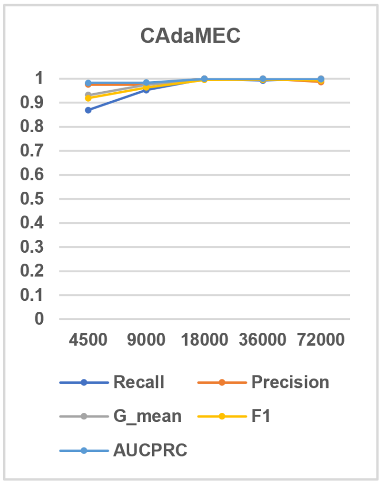

Figure 20. CAdaMEC: Impact of Dataset Size

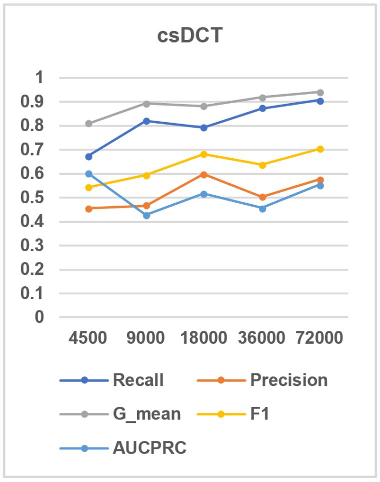

Figure 21. csDCT: Impact of Dataset Size

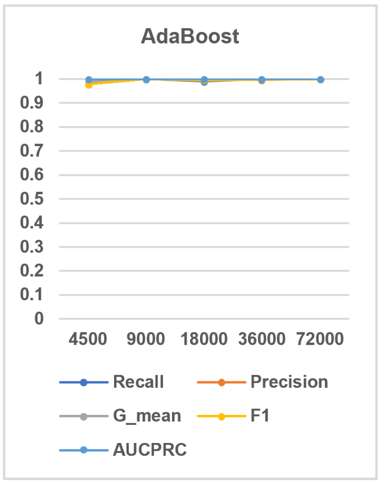

Figure 22. AdaBoost: Impact of Dataset Size

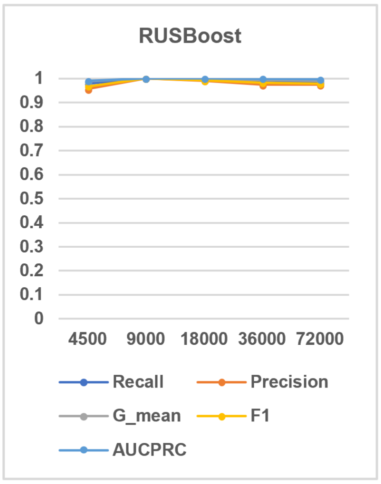

Figure 23. RUSBoost: Impact of Dataset Size

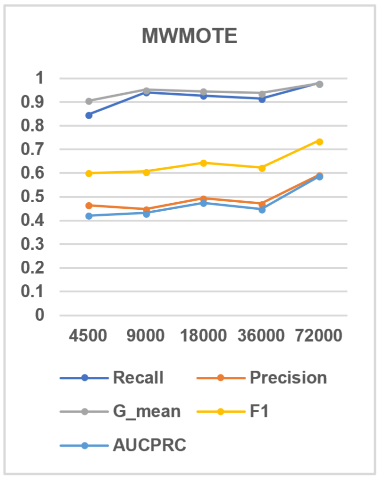

Figure 24. MWMOTE: Impact of Dataset Size

a laptop with Intel(R) Core(TM) i7-7660U CPU @ 2.50GHz and 16GB RAM, running Windows 10 Version 20H2.

Table 23 shows the learning time achieved by each algorithm on different datasets. IML and DDAE yield the longest learning time compared to the other

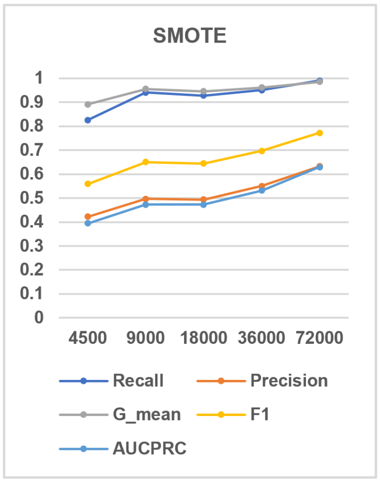

Figure 25. SMOTE: Impact of Dataset Size

algorithms. Self-paced Ensemble Classifier performs fairly well in terms of learning time with most of the values ranging in (0, 0.1).

As shown in Table 24, the memory usage of each algorithm is similar, even when processing different datasets, and there is no obvious relation between the memory usage and the size of the dataset. Among all these algorithms, AdaBoost and self-paced Ensemble Classifier are two algorithms with the lowest memory usage.

5. Discussion

As shown in our experiments, DDAE yields the highest value for recall on almost all imbalanced datasets. Since the recall only focuses on minority, a higher recall means that most samples with a class label of 1 can be correctly predicted. since the recall and precision generally cannot be satisfactory at the same time, 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. 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 Fig. 15 and Fig. 16, 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 consistently performs well in F1 and AUCPRC, even when the value of recall is low. compared with other models, the evaluation metrics of IML on different datasets vary, meaning that IML is more sensitive to the data distribution of the dataset. The results in Section 4.2 shows the performance of IML keeps stable under different IRs.

The performance of SMOTE and MWMOTE seems quite similar, but the G-mean, F1 and AUCPRC for SMOTE on some slightly imbalanced datasets, such as WDBC (IR=1.866) and PID (IR=1.684), are better than that of MWMOTE. This can also be observed from Fig. 14, which shows that the IR is relatively small, SMOTE performs better than MWMOTE. SMOTE will randomly select minority samples to synthesize new samples, regardless of the surrounding samples. This may generate useless samples if the selected minority sample is far away from the decision boundary; the newly generated samples may be also overlapped with the majority samples in the surrounding if the selected minority resides inside the majority region [52]. 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 [42]. Different from SMOTE, MWMOTE employs a more practical approach to select samples that are difficult to learn, and then assigns them appropriate weights [42]. This can explain why MWMOTE performs better than SMOTE when the IR is high.

AdaBoost, RUSBoost and self-paced Ensemble Classifier are three alternative ensemble learning techniques to DDAE. 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. Unlike DDAE, the other three ensemble methods do not take the weights of both classes into account, which can also contribute to a lower recall value but higher F1 and AUCPRC values. AdaBoost and RUSBoost perform similarly. Compared with RUSBoost, the self-paced Ensemble Classifier and AdaBoost show a more stable performance. Even though AdaBoost and RUSBoost and AdaBoost achieve 0.196 and 0.214 respectively in terms of recall on the Pc3 dataset, the recall of the self-paced Ensemble Classifier is 0.786 for

Table 23. Learning Time for All Models on the Public Datasets

Dataset	Learning Time (in Seconds)									
	DDAE	**MWMOTE**	**SMOTE**	**RUSBoost**	**AdaBoost**	**MetaCost**	**csDCT**	**CAdaMEC**	**self-paced**	**IML**
Euthyroid Sick	146.78	0.381	0.224	0.050	0.110	2.426	7.586	0.185	**0.041**	912.87
Thyroid Sick	471.88	1.362	1.711	0.542	1.103	19.910	16.786	0.370	**0.080**	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	637.39	0.983	3.207	0.863	0.976	31.625	166.284	0.372	**0.070**	5001.33
MGC	1472.36	0.845	0.749	0.529	0.679	54.577	31.973	0.231	**0.065**	9861.93
WDBC	13.43	0.950	0.960	**0.864**	0.894	0.818	0.909	16.117	**0.864**	8.11
PID	17.45	0.381	0.224	0.381	0.478	9.187	15.547	0.089	**0.029**	27.03
Yeast1vs7	46.99	0.083	**0.014**	0.235	0.141	2.411	11.044	0.109	0.036	5.85
Cm1	54.64	0.069	0.060	**0.016**	0.106	1.201	36.415	0.086	0.029	10.87
Pc1	76.46	0.103	0.119	0.244	0.177	3.570	35.073	0.106	**0.032**	8.70
Pc3	110.98	0.358	0.053	0.274	0.258	5.388	136.251	0.157	**0.034**	180.43
Pc4	97.79	0.399	0.638	0.259	0.208	5.175	65.156	0.118	**0.038**	156.00
Poker8vs6	126.93	0.101	**0.034**	0.185	0.135	4.565	7.267	0.090	0.038	134.87
Poker89vs6	101.16	0.136	**0.020**	0.236	0.261	4.617	6.322	0.102	0.029	135.84

Table 24. Memory Usage for All Models on the Public Datasets

Dataset	Memory Usage (in MB)									
	DDAE	**MWMOTE**	**SMOTE**	**RUSBoost**	**AdaBoost**	**MetaCost**	**csDCT**	**CAdaMEC**	**self-paced**	**IML**
Euthyroid Sick	1.587	0.833	0.388	0.417	**0.342**	0.418	1.449	0.369	0.399	0.977
Thyroid Sick	1.978	0.425	0.362	0.403	**0.344**	0.410	0.956	0.364	0.354	0.481
PH1	1.964	0.428	0.356	0.391	**0.338**	0.400	1.149	0.357	0.348	0.592
PH2	1.986	0.817	0.369	0.417	**0.350**	0.417	1.523	0.351	0.366	0.607
optdigits	1.677	0.416	0.439	0.344	**0.330**	0.394	0.813	0.336	0.336	0.413
MGC	1.996	0.822	0.885	0.425	**0.347**	0.762	1.055	0.376	0.353	0.457
WDBC	1.609	0.549	0.451	0.677	0.390	**0.388**	1.414	0.725	0.447	0.619
PID	0.707	0.457	0.450	0.355	**0.338**	0.383	0.810	0.343	0.346	0.426
Yeast1vs7	0.430	0.438	0.428	0.354	**0.334**	0.396	0.781	0.342	0.358	0.504
Cm1	1.499	0.614	0.515	0.399	**0.343**	0.410	0.829	0.366	0.358	0.437
Pc1	1.966	0.549	0.451	0.677	**0.390**	0.388	1.410	0.725	0.447	0.619
Pc3	1.503	0.530	0.458	0.793	**0.361**	0.417	1.305	0.372	0.381	0.612
Pc4	1.007	0.474	0.429	0.366	**0.342**	0.387	0.808	0.348	0.348	0.434
Poker8vs6	1.510	0.525	0.464	0.392	**0.344**	0.408	0.911	0.367	0.365	0.435
Poker89vs6	0.638	0.404	0.418	0.360	**0.336**	0.395	0.836	0.346	0.346	0.426

this dataset, which yields the highest value among all our investigated models.

Moreover, the cost-sensitive learning methods (MetaCost, csDCT and CAdaMEC) 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. Our experiments confirmed that as the IR 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 shows an irregular fluctuation), the remaining algorithms either consistently 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 resampling methods, such as in MWMOTE and SMOTE.

Although our current experiments focus on binary classification, some of the investigated algorithms are also suitable for multi-class classification, such as AdaBoost, MetaCost and cost-sensitive decision tree.

SMOTE, MWMOTE and IML focus on the feature space. Thus, if these techniques are used for multi-class classification, the classifier combined with these techniques should be also suitable for multi-class classification.

Table 25 summarizes the pros and cons of all the techniques investigated in this paper. As a consequence, 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 precision. However, in medical diagnosis or earthquake prediction fields, the recall for the algorithms is more significant.

Based on our experimental results in the studied datasets, we recommend the following flowchart for algorithm selection, as shown in Fig. 26.

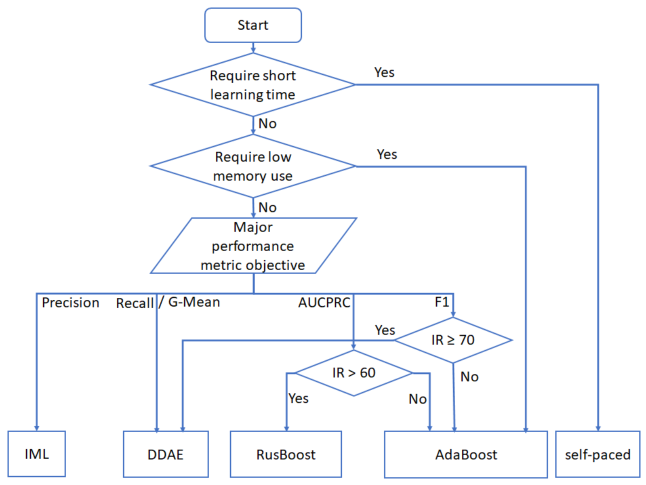

Figure 26. Our recommendation for algorithm selection based on experimental results in this paper

6. Conclusion

We conducted a systematic performance comparison of several class imbalance classification models using various datasets from medical and other sectors.

Our experiments showed sampling methods perform the worst, and when the dataset is not very imbalanced cost-sensitive learning models achieves good performance. Ensemble learning techniques generally perform better over other approaches due to their combined intelligence of multiple basic classifiers. In terms of system performance, distance metric learning requires the longest learning time, while self-paced Ensemble Classifier requires shortest time for learning; AdaBoost and self-paced Ensemble Classifier requires the lowest memory usage.

Our analysis implies that the performance of algorithms cannot be judged by the individual evaluation metrics alone; the application requirements and usage scenarios should also be taken into account. In particular, we make our empirical recommendation for algorithm selection under different requirements and usage scenarios. In addition, the number of base estimators set by ensemble learning classifiers can also make an impact on their performance.

Acknowledgement

This research was funded by the Fundamental Research Funds for the Central Universities (Grant No. 2020JJ014, YY19SSK05). The authors would like to acknowledge Yachao Shao, Xiaoming Fu and anonymous reviewers for their constructive comments and suggestions.

References

[1] Sun, X., Wang, H., Li, J. and Pei, J. (2011) Publishing anonymous survey rating data. *Data Mining and Knowledge Discovery* **23**(3): 379–406.

[2] Li, H., Wang, Y., Wang, H. and Zhou, B. (2017) Multi-window based ensemble learning for classification of imbalanced streaming data. *World Wide Web* **20**(6): 1507–1525.

[3] Sarki, R., Ahmed, K., Wang, H. and Zhang, Y. (2020) Automated detection of mild and multi-class diabetic eye diseases using deep learning. *Health Information Science and Systems* **8**(1): 1–9.

[4] He, H. and Ma, Y. (2013) *Imbalanced Learning: Foundations, Algorithms, and Applications* (Wiley).

[5] Zheng, Z., Cai, Y. and Li, Y. (2016) Oversampling method for imbalanced classification. *Computing and Informatics* **34**(5): 1017–1037.

[6] Castro, C.L. and Braga, A.P. (2013) Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data. *IEEE transactions on Neural Networks and Learning Systems* **24**(6): 888–899.

[7] Cover, T.M. and Hart, P.E. (1967) Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* **13**(1): 21–27.

[8] Cortes, C. and Vapnik, V. (1995) Support-vector networks. *Machine Learning* **20**(3): 273–297.

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

[10] Breiman, L. (2001) Random forests. *Machine Learning* **45**(1): 5–32.

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

[12] Hansen, L.K. and Salamon, P. (1990) Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence* **12**(10): 993–1001.

[13] Du, J., Michalska, S., Subramani, S., Wang, H. and Zhang, Y. (2019) Neural attention with character embeddings for hay fever detection from twitter. *Health information science and systems* **7**(1): 1–7.

Table 25. Pros and cons of different techniques utilized for dealing with class imbalance problem

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	RUSBoost DDAE
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
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 DDAE
Ensemble Learning	Stable; Make more reliable predictions than the single classifier; Reduce the dispersion of model performance	Inherit both advantages and disadvantages of used techniques when the final classifier is combined with several techniques	RUSBoost AdaBoost self-paced Ensemble Classifier DDAE

[14] Wang, L., Zhao, Z., Luo, Y., Yu, H., Wu, S., Ren, X., Zheng, C. *et al.* (2020) Classifying 2-year recurrence in patients with dlbcl using clinical variables with imbalanced data and machine learning methods. *Computer Methods and Programs in Biomedicine* **196**: 105567. doi:https://doi.org/10.1016/j.cmpb.2020.105567, URL https://www.sciencedirect.com/science/article/pii/S0169260719320024.

[15] Singh, R., Zhang, Y., Wang, H., Miao, Y. and Ahmed, K. (2020) Investigation of social behaviour patterns using location-based data – a Melbourne case study. *EAI Endorsed Transactions on Scalable Information Systems: Online First* doi:10.4108/eai.26-10-2020.166767.

[16] Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L. and Herrera, F. (2011) KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing* **17**(2-3): 255–287.

[17] Pandey, S.K., Mishra, R.B. and Tripathi, A.K. (2021) Machine learning based methods for software fault prediction: A survey. *Expert Systems with Applications* **172**: 114595. doi:https://doi.org/10.1016/j.eswa.2021.114595, URL https://www.sciencedirect.com/science/article/pii/S0957417421000361.

[18] Lake, R. (2020), 23 frightening credit card fraud statistics, https://www.creditdonkey.com/credit-card-fraud-statistics.html.

[19] García, V., Marqués, A.I. and Sánchez, J.S. (2019) Exploring the synergetic effects of sample types on the performance of ensembles for credit risk and corporate bankruptcy prediction. *Information Fusion* **47**: 88–101. doi:https://doi.org/10.1016/j.inffus.2018.07.004, URL https://www.sciencedirect.com/science/article/pii/S1566253517308011.

[20] Al-Hashedi, K.G. and Magalingam, P. (2021) Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019. *Computer Science Review* **40**: 100402. doi:https://doi.org/10.1016/j.cosrev.2021.100402, URL https://www.sciencedirect.com/science/article/pii/S1574013721000423.

[21] Zhang, J., Chen, H. and Fu, X. (2021 (accepted)) CPFinder: Finding an Unknown Caller's Profession from Anonymized Mobile Phone Data. *Digital Communications and Networks* .

[22] Understanding disaster risk data, https://www.preventionweb.net/risk/datasets.

[23] Devaraj, A., Murthy, D. and Dontula, A. (2020) Machine-learning methods for identifying social media-based requests for urgent help during hurricanes. *International Journal of Disaster Risk Reduction* **51**: 101757. doi:https://doi.org/10.1016/j.ijdrr.2020.101757, URL https://www.sciencedirect.com/science/article/pii/S2212420920312590.

[24] Rangarajan, S. and Chandar, R. (2017) Qos-based architecture for discovery and selection of suitable web services using non-functional properties. *EAI Endorsed Transactions on Scalable Information Systems* **4**(12).

[25] Peng, M., Zeng, G., Sun, Z., Huang, J., Wang, H. and Tian, G. (2018) Personalized app recommendation based on app permissions. *World Wide Web* **21**(1): 89–104.

[26] Zeng, M., Zou, B., Wei, F., Liu, X. and Wang, L. (2016) 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): 225–228.

[27] Padurariu, C. and Breaban, M. (2019) Dealing with data imbalance in text classification. *Procedia Computer Science* **159**: 736–745. doi:10.1016/j.procs.2019.09.229.

[28] Glazkova, A., A comparison of synthetic oversampling methods for multi-class text classification, arXiv preprint arXiv:2008.04636.

[29] Elkan, C. (2001) The foundations of cost-sensitive learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, **17**: 973–978.

[30] Sammut, C. and Webb, G.I. (2011) *Encyclopedia of Machine Learning* (Boston, MA: Springer).

[31] Dudjak, M. and Martinović, G. (2021) An empirical study of data intrinsic characteristics that make learning from imbalanced data difficult. *Expert Systems with Applications* **182**: 115297. doi:https://doi.org/10.1016/j.eswa.2021.115297, URL https://www.sciencedirect.com/science/article/pii/S0957417421007272.

[32] Orriols-Puig, A. and Bernadó-Mansilla, E. (2009) Evolutionary rule-based systems for imbalanced data sets. *Soft Computing* **13**(3): 213–225.

[33] Chawla, N.V., Japkowicz, N. and Kotcz, A. (2004) Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter* **6**(1): 1–6.

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

[35] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P. (2002) SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* **16**: 321–357.

[36] Wang, N., Zhao, X., Jiang, Y. and Gao, Y. (2018) Iterative metric learning for imbalance data classification. In *IJCAI*: 2805–2811.

[37] Zhou, Z.H. (2012) *Ensemble methods: foundations and algorithms* (Boca Raton, FL: CRC Press).

[38] Fan, W., Stolfo, S.J., Zhang, J. and Chan, P.K. (1999) Adacost: misclassification cost-sensitive boosting. In *ICML*, **99**: 97–105.

[39] Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J. and Napolitano, A. (2009) RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* **40**(1): 185–197.

[40] Yin, J., Gan, C., Zhao, K., Lin, X., Quan, Z. and Wang, Z.J. (2020) A novel model for imbalanced data classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, **34**: 6680–6687.

[41] Saito, T. and Rehmsmeier, M. (2015) The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS One* .

[42] Barua, S., Islam, M.M., Yao, X. and Murase, K. (2012) MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering* **26**(2): 405–425.

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

[44] Nikolaou, N., Edakunni, N., Kull, M., Flach, P. and Brown, G. (2016) Cost-sensitive boosting algorithms: Do we really need them? *Machine Learning* **104**(2): 359–384.

[45] Freund, Y. and Schapire, R.E. (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* **55**(1): 119–139.

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

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

[48] Liu, N., Li, X., Qi, E., Xu, M., Li, L. and Gao, B. (2020) A novel ensemble learning paradigm for medical diagnosis with imbalanced data. *IEEE Access* **8**: 171263–171280.

[49] Liu, X.Y., Wu, J. and Zhou, Z.H. (2008) Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **39**(2): 539–550.

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

[51] He, H. and Ma, Y. (2013) Imbalanced learning: foundations, algorithms, and applications .

[52] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P. (2002) SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* **16**: 321–357.

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

[54] He, H. and Garcia, E.A. (2009) Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* **21**(9): 1263–1284.

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

[56] Ling, C.X. and Sheng, V.S. (2008) Cost-sensitive learning and the class imbalance problem. *Encyclopedia of Machine Learning* **2011**: 231–235.

[57] Shao, Y., Zhao, T., Wang, X., Zou, X. and Fu, X. (2020) Multiple balance subsets stacking for imbalanced healthcare datasets. In *Proceedings of IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS 2020)*.

[58] Sahin, Y., Bulkan, S. and Duman, E. (2013) A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications* **40**(15): 5916–5923.

[59] Ting, K.M. (1998) Inducing cost-sensitive trees via instance weighting. In *European Symposium on Principles of Data Mining and Knowledge Discovery* (Springer): 139–147.

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

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

[62] Gautheron, L., Habrard, A., Morvant, E. and Sebban, M. (2020) Metric learning from imbalanced data with generalization guarantees. *Pattern Recognition Letters* **133**: 298–304. doi:https://doi.org/10.1016/j.patrec.2020.03.008, URL https://www.sciencedirect.com/science/article/

pii/S0167865520300866.

[63] Dasarathy, B.V. and Sheela, B.V. (1979) A composite classifier system design: Concepts and methodology. *Proceedings of the IEEE* **67**(5): 708–713.

[64] Rokach, L. (2010) *Pattern classification using ensemble methods*, **75** (Hackensack, NJ: World Scientific Publishing).

[65] Han, L., Luo, S., Yu, J., Pan, L. and Chen, S. (2014) Rule extraction from support vector machines using ensemble learning approach: an application for diagnosis of diabetes. *IEEE Journal of Biomedical and Health Informatics* **19**(2): 728–734.

[66] Wang, Y.C. and Cheng, C.H. (2021) A multiple combined method for rebalancing medical data with class imbalances. *Computers in Biology and Medicine* **134**: 104527. doi:https://doi.org/10.1016/j.compbiomed.2021.104527, URL https://www.sciencedirect.com/science/article/pii/S0010482521003218.

[67] Merkwirth, C., Mauser, H., Schulz-Gasch, T., Roche, O., Stahl, M. and Lengauer, T. (2004) Ensemble methods for classification in cheminformatics. *Journal of Chemical Information and Computer Sciences* **44**(6): 1971–1978.

[68] Yin, Z., Ai, H., Zhang, L., Ren, G., Wang, Y., Zhao, Q. and Liu, H. (2019) Predicting the cytotoxicity of chemicals using ensemble learning methods and molecular fingerprints. *J Appl Toxicol* **39**(10): 1366–1377.

[69] Yang, P., Hwa Yang, Y., B Zhou, B. and Y Zomaya, A. (2010) A review of ensemble methods in bioinformatics. *Current Bioinformatics* **5**(4): 296–308.

[70] Wan, S., Duan, Y. and Q, Z. (2017) HPSLPred: An ensemble multi-label classifier for human protein subcellular location prediction with imbalanced source. *Proteomics* **17**(17-18).

[71] Schapire, R.E. (1990) The strength of weak learnability. *Machine Learning* **5**(2): 197–227.

[72] Polikar, R. (2006) Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine* **6**(3): 21–45.

[73] Freund, Y. and Schapire, R.E. (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* **55**(1): 119–139.

[74] Nikolaou, N. and Brown, G. (2015) Calibrating AdaBoost for asymmetric learning. In *International Workshop on Multiple Classifier Systems* (Springer): 112–124.

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

[76] Sun, Y., Wong, A.K. and Kamel, M.S. (2009) Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence* **23**(04): 687–719.

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

[78] Ferri, C., Hernández-Orallo, J. and Modroiu, R. (2009) An experimental comparison of performance measures for classification. *Pattern Recognition Letters* **30**(1): 27–38.

[79] Hossin, M. and Sulaiman, M. (2015) A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process* **5**(2): 1.

[80] Guo, X., Yin, Y., Dong, C., Yang, G. and Zhou, G. (2008) On the class imbalance problem. In *2008 Fourth International Conference on Natural Computation*, **4**: 192–201. doi:10.1109/ICNC.2008.871.

[81] Su, C.T. and Hsiao, Y.H. (2007) An evaluation of the robustness of MTS for imbalanced data. *IEEE Transactions on Knowledge and Data Engineering* **19**(10): 1321–1332.

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

[83] Kubat, M., Holte, R. and Matwin, S. (1997) Learning when negative examples abound. In *European Conference on Machine Learning* (Springer): 146–153.

[84] Nguyen, G.H., Bouzerdoum, A. and Phung, S.L. (2009) Learning pattern classification tasks with imbalanced data sets. *Pattern Recognition* : 193–208.

[85] Ballabio, D., Grisoni, F. and Todeschini, R. (2018) Multivariate comparison of classification performance measures. *Chemometrics and Intelligent Laboratory Systems* **174**: 33–44.

[86] Fawcett, T. (2006) An introduction to ROC analysis. *Pattern Recognition Letters* **27**(8): 861–874.

[87] Davis, J. and Goadrich, M. (2006) The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning*: 233–240.

[88] Richardson, M. and Domingos, P. (2006) Markov logic networks. *Machine Learning* **62**(1-2): 107–136.

[89] Waspada, I., Bahtiar, N., Wirawan, P.W. and Awan, B.D.A. (2020) Performance analysis of isolation forest algorithm in fraud detection of credit card transactions. *Khazanah Informatika: Jurnal Ilmu Komputer dan Informatika* **6**(2).

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

[91] Dua, D. and Graff, C. (2017), Uci machine learning repository, http://archive.ics.uci.edu/ml.

[92] 2004, K.C., Particle physics; plus protein homology prediction, https://www.kdd.org/kdd-cup/view/kdd-cup-2004/Data. 2004.

[93] Menzies, T., Greenwald, J. and Frank, A. (2006) Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering* **33**(1): 2–13.

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

[95] Han, J., Kamber, M. and Pei, J. (2011) Data mining concepts and techniques (third edition). *The Morgan Kaufmann Series in Data Management Systems* **5**(4): 83–124.

[96] Kotsiantis, S.B., Kanellopoulos, D. and Pintelas, P.E. (2006) Data preprocessing for supervised leaning. *International Journal of Computer Science* **1**(2): 111–117.

[97] Kamiran, F. and Calders, T. (2012) Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems* **33**(1): 1–33.

[98] Allison, P.D. (2001) *Missing data* (Sage Publications).

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